Probabilistic Programming and its Applications

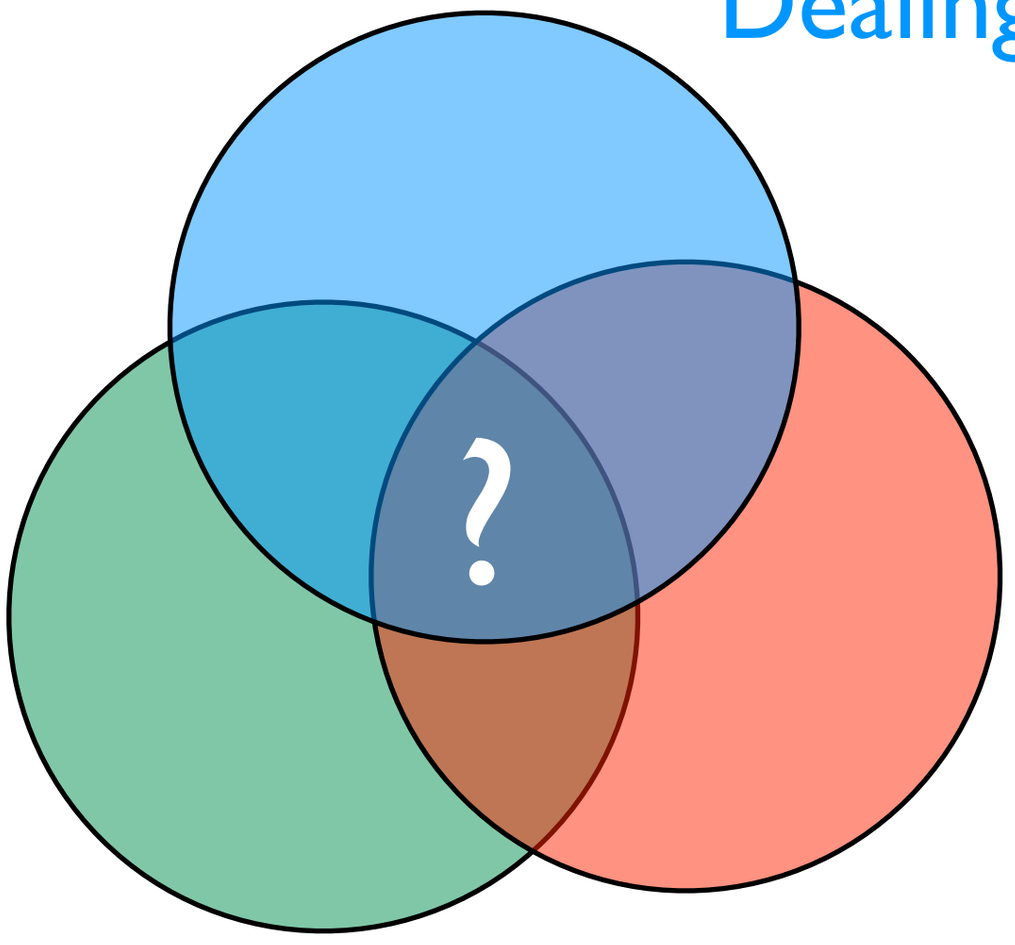
Luc De Raedt

KI 2017

A key question in AI:

Dealing with uncertainty

Reasoning with relational data



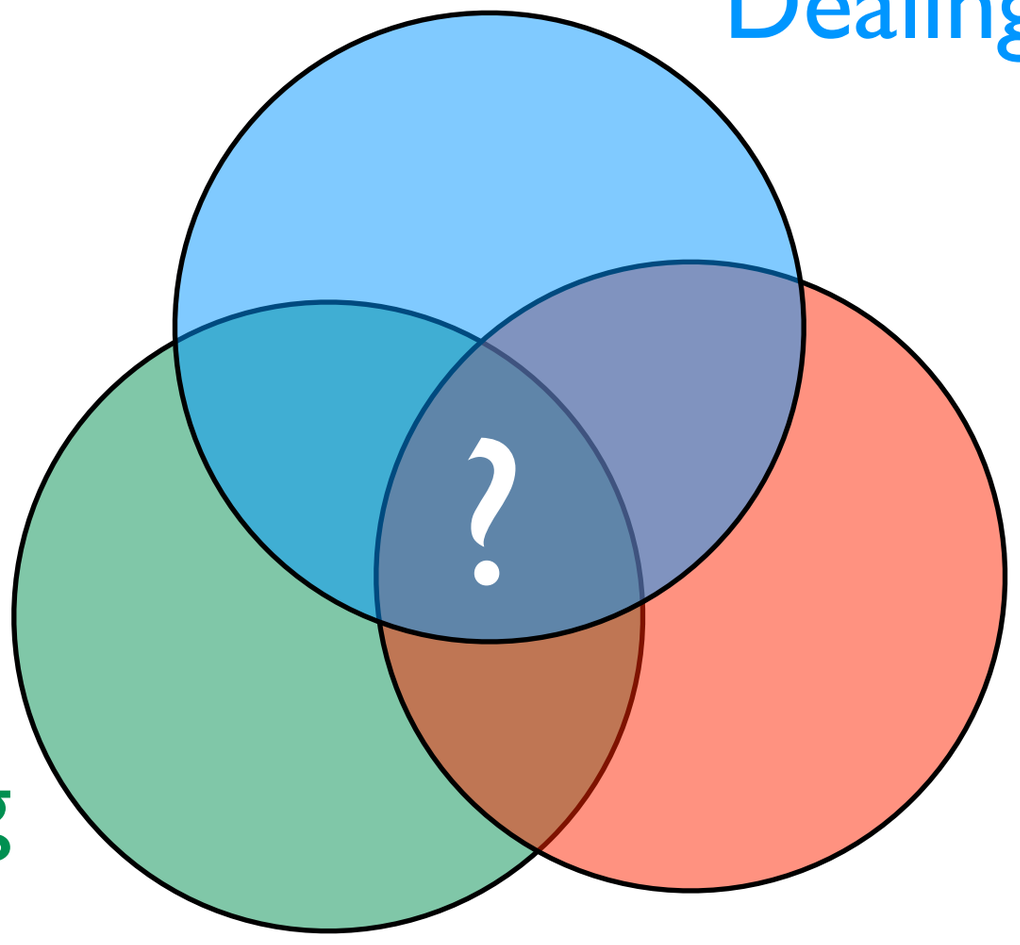
Learning

A key question in AI:

Dealing with uncertainty

Reasoning with relational data

- logic
- databases
- programming
- ...

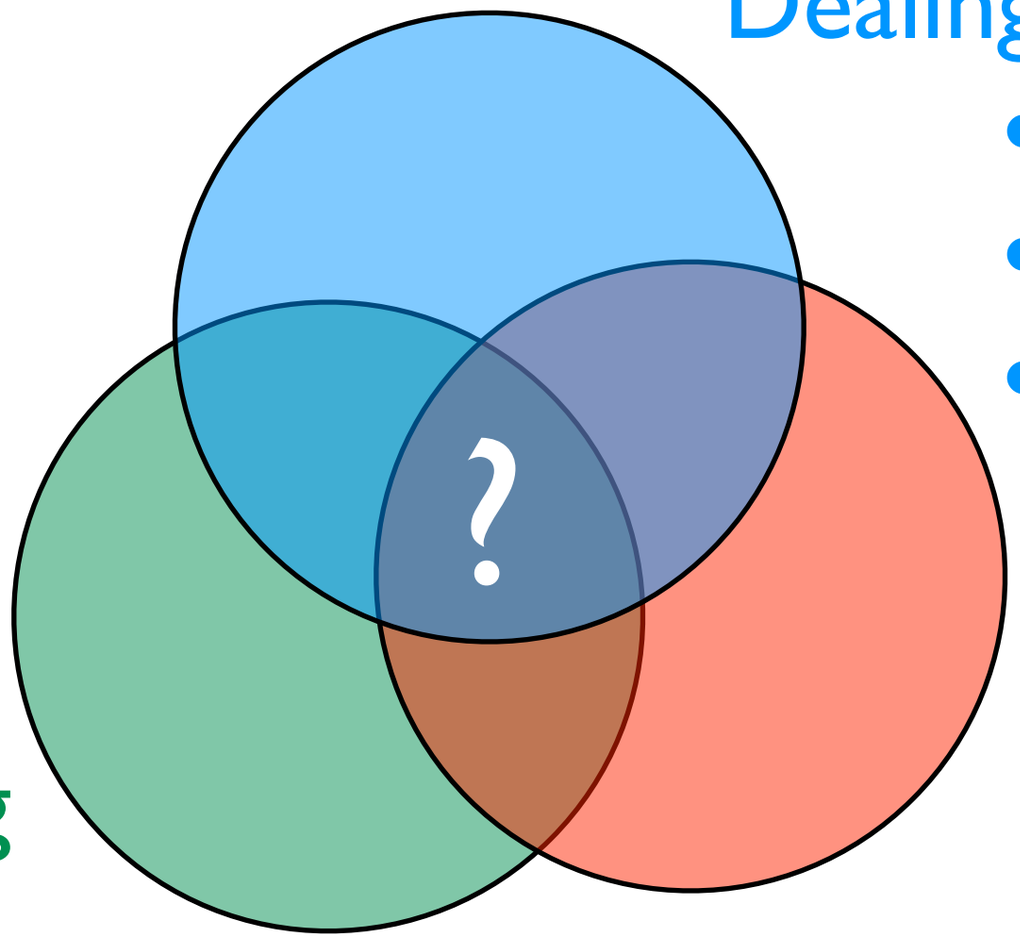


Learning

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

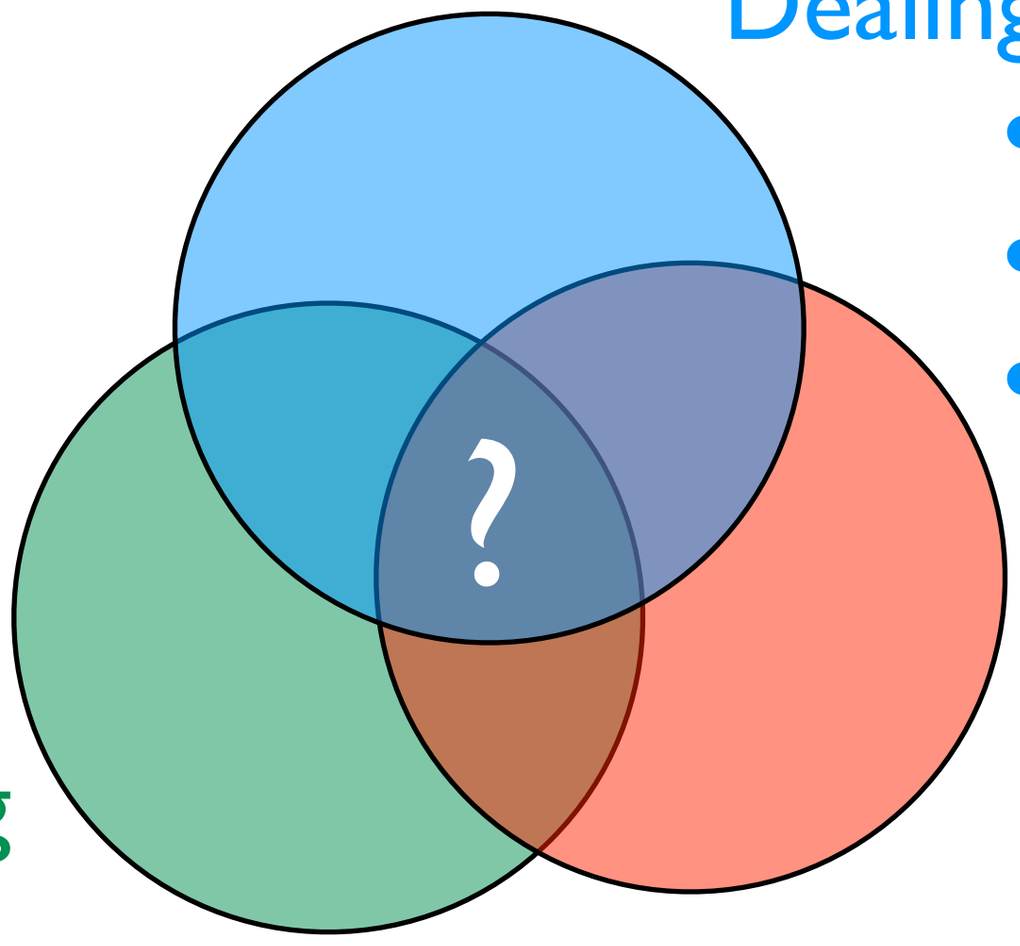
- probability theory
- graphical models
- ...

Learning

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

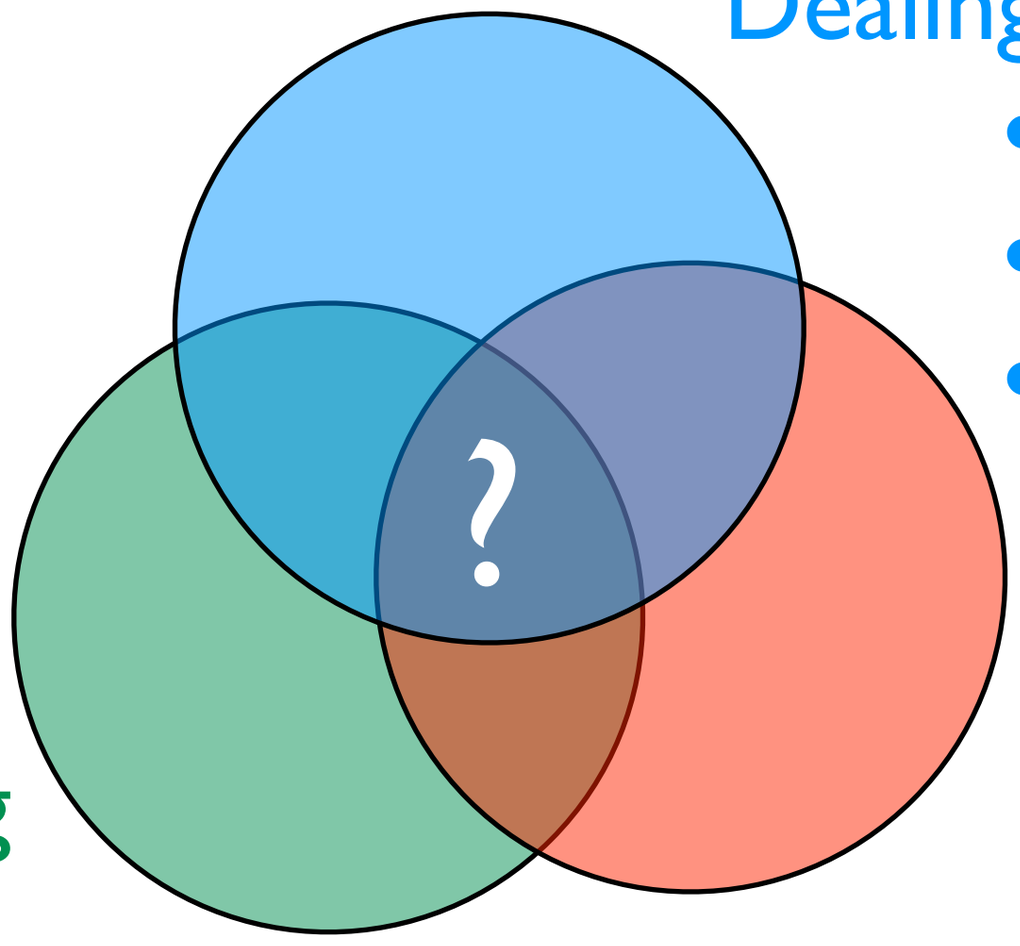
Learning

- parameters
- structure

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Example: Information Extraction

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

NELL - Never Ending Language Learning:
http://rtw.ml.cmu.edu/rtw/

Example: Information Extraction

Recently-Learned Facts twitter

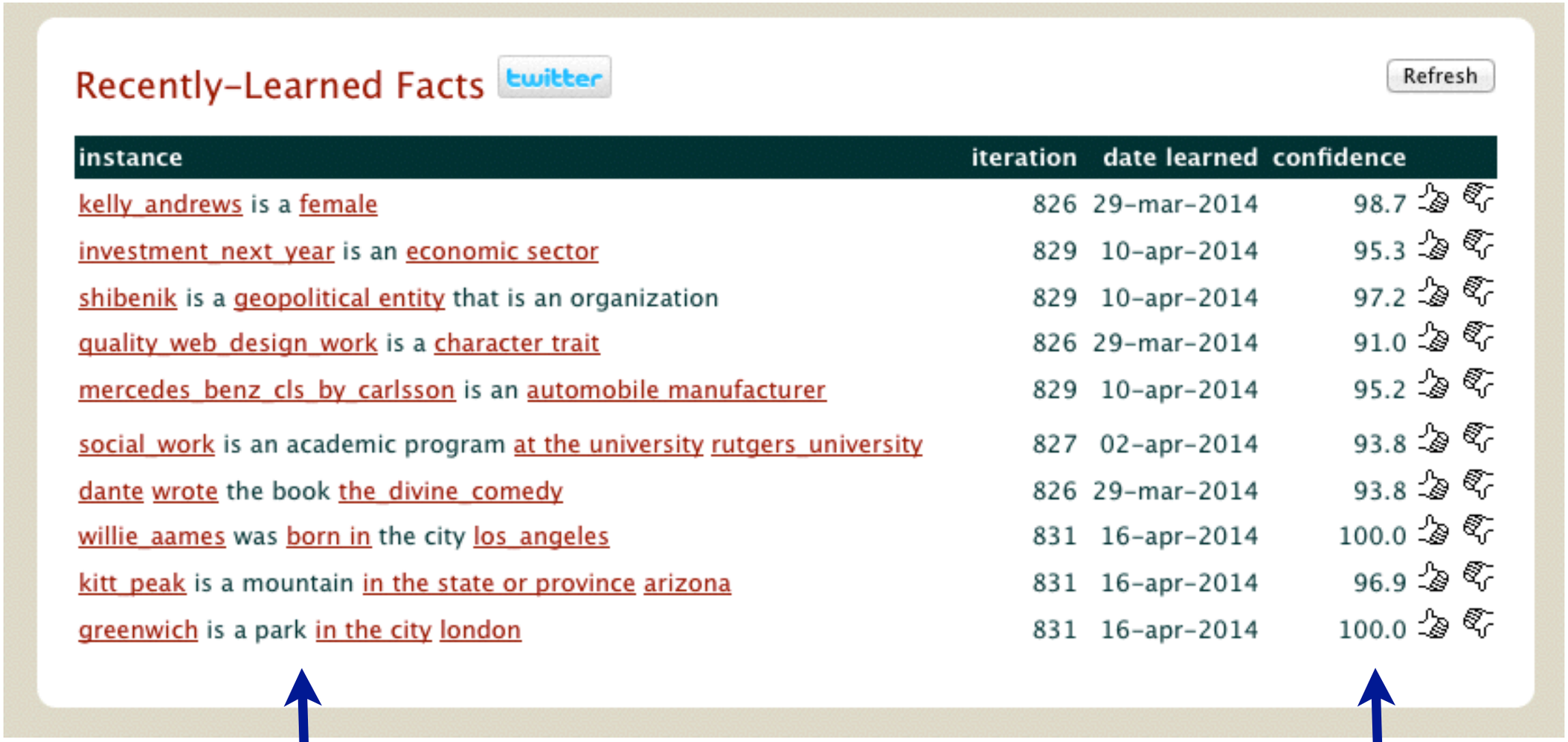
Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

NELL - Never Ending Language Learning: http://rtw.ml.cmu.edu/rtw/

Example: Information Extraction

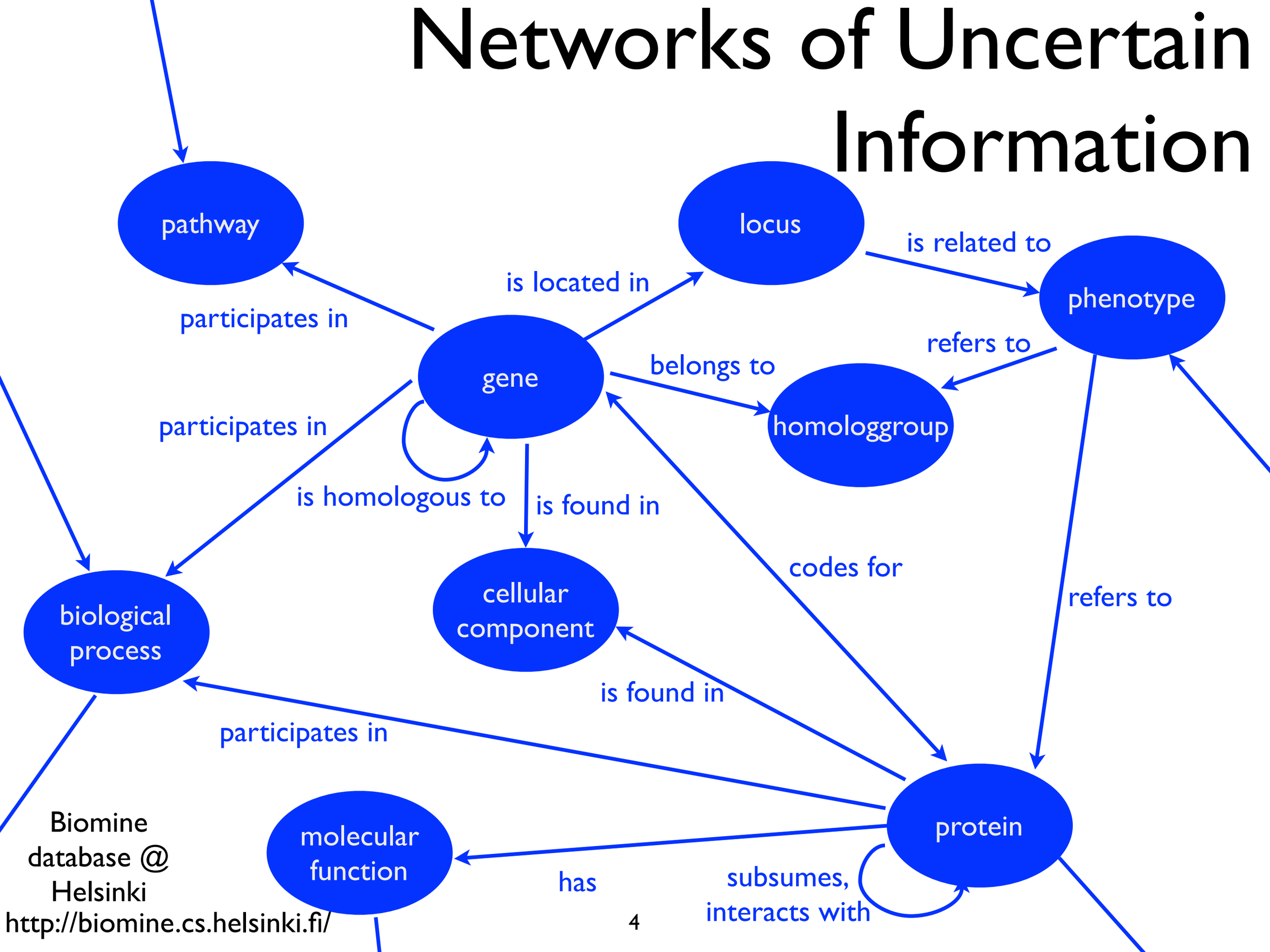


instances for many different relations

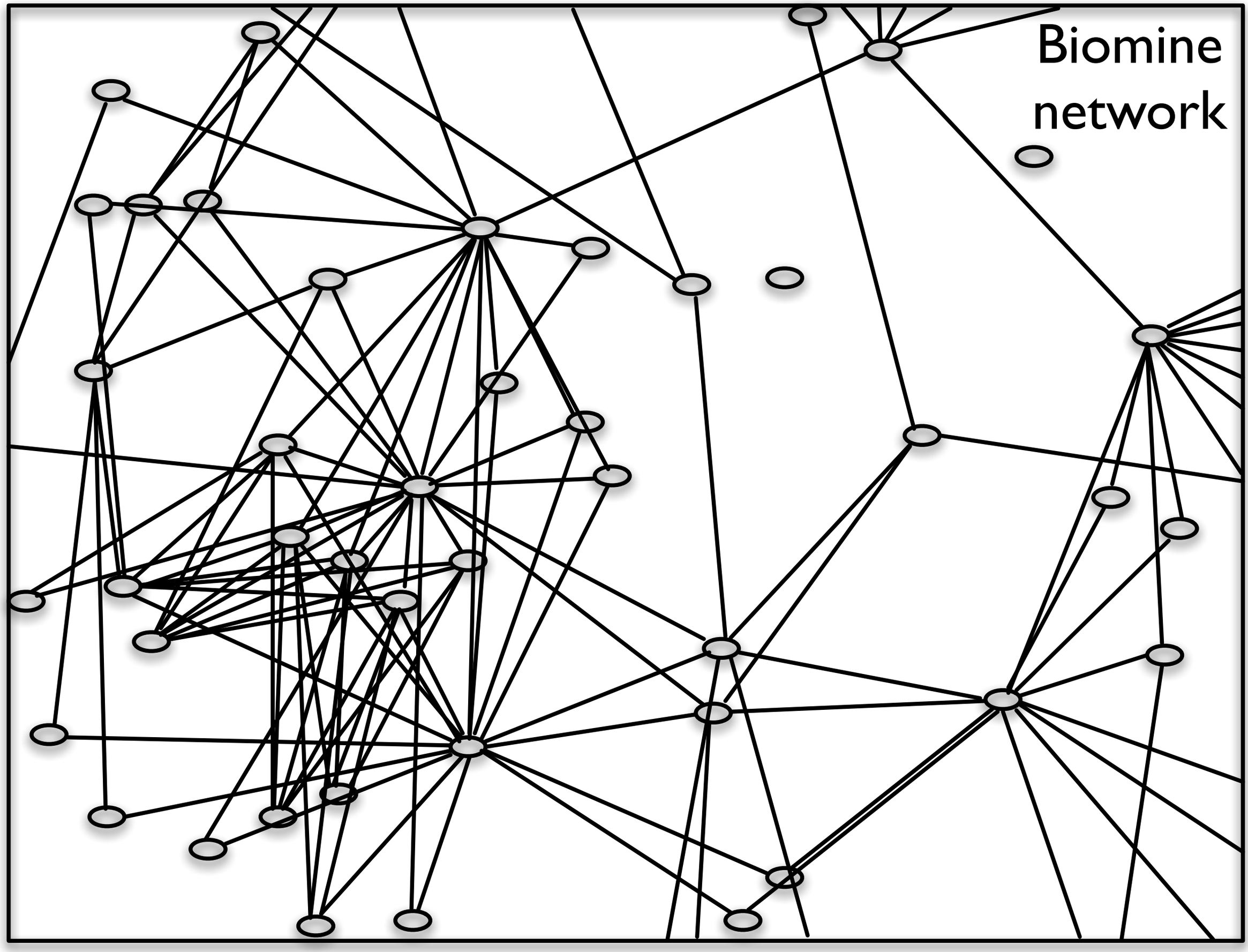
degree of certainty

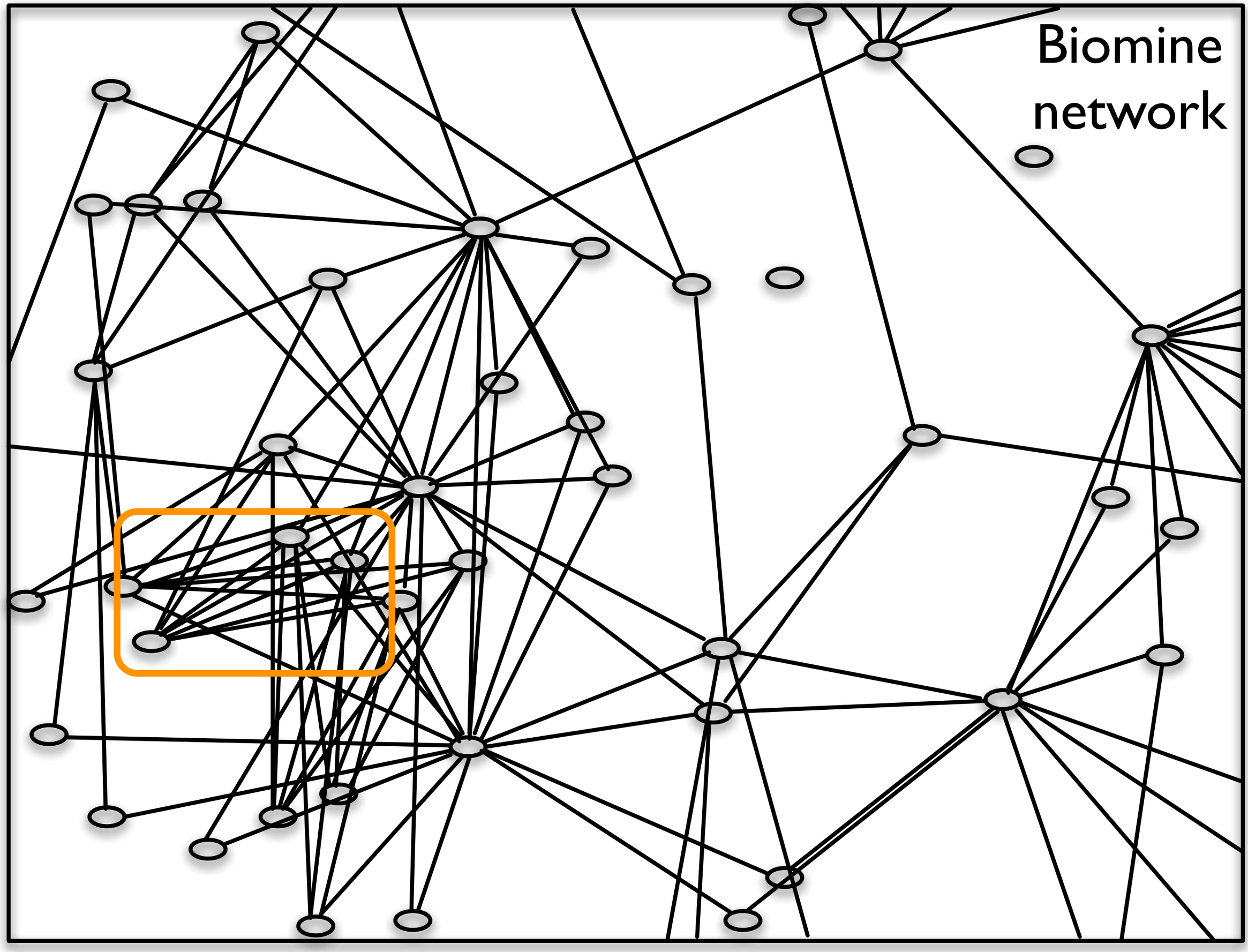
NELL - Never Ending Language Learning: http://rtw.ml.cmu.edu/rtw/

Networks of Uncertain Information

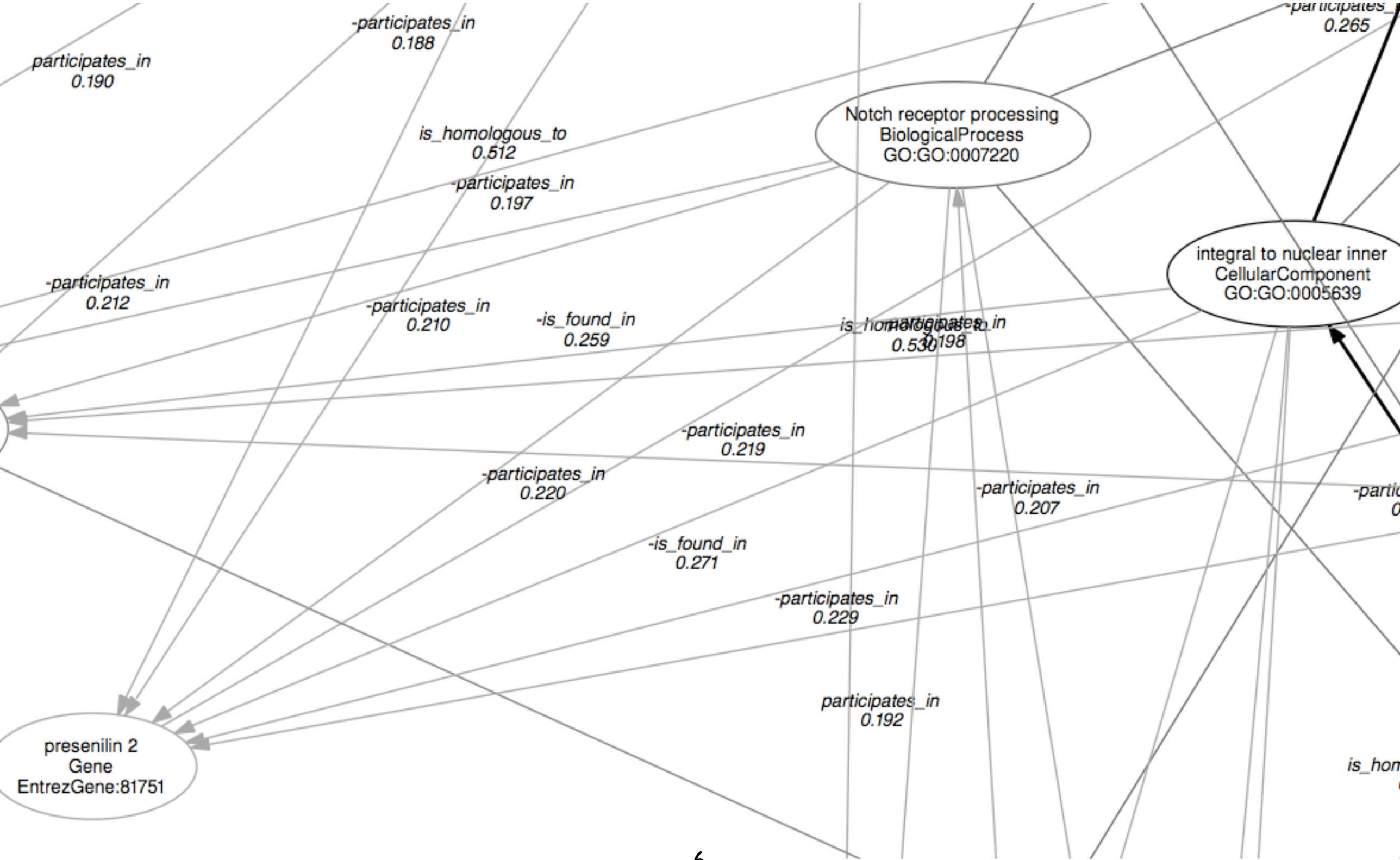


Biomine database @ Helsinki
http://biomine.cs.helsinki.fi/


Biomine
network

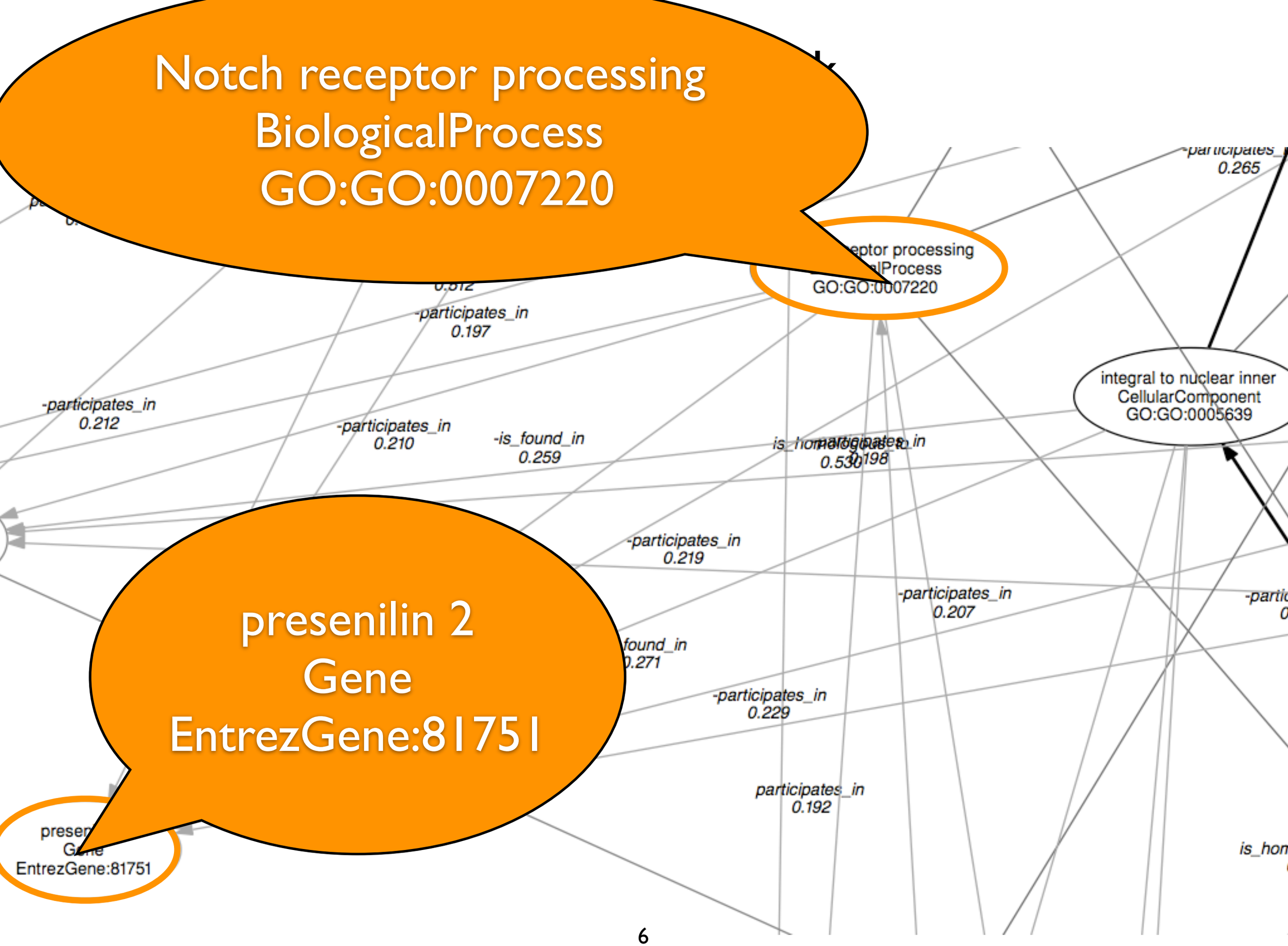

Biomine
network

Biomine Network

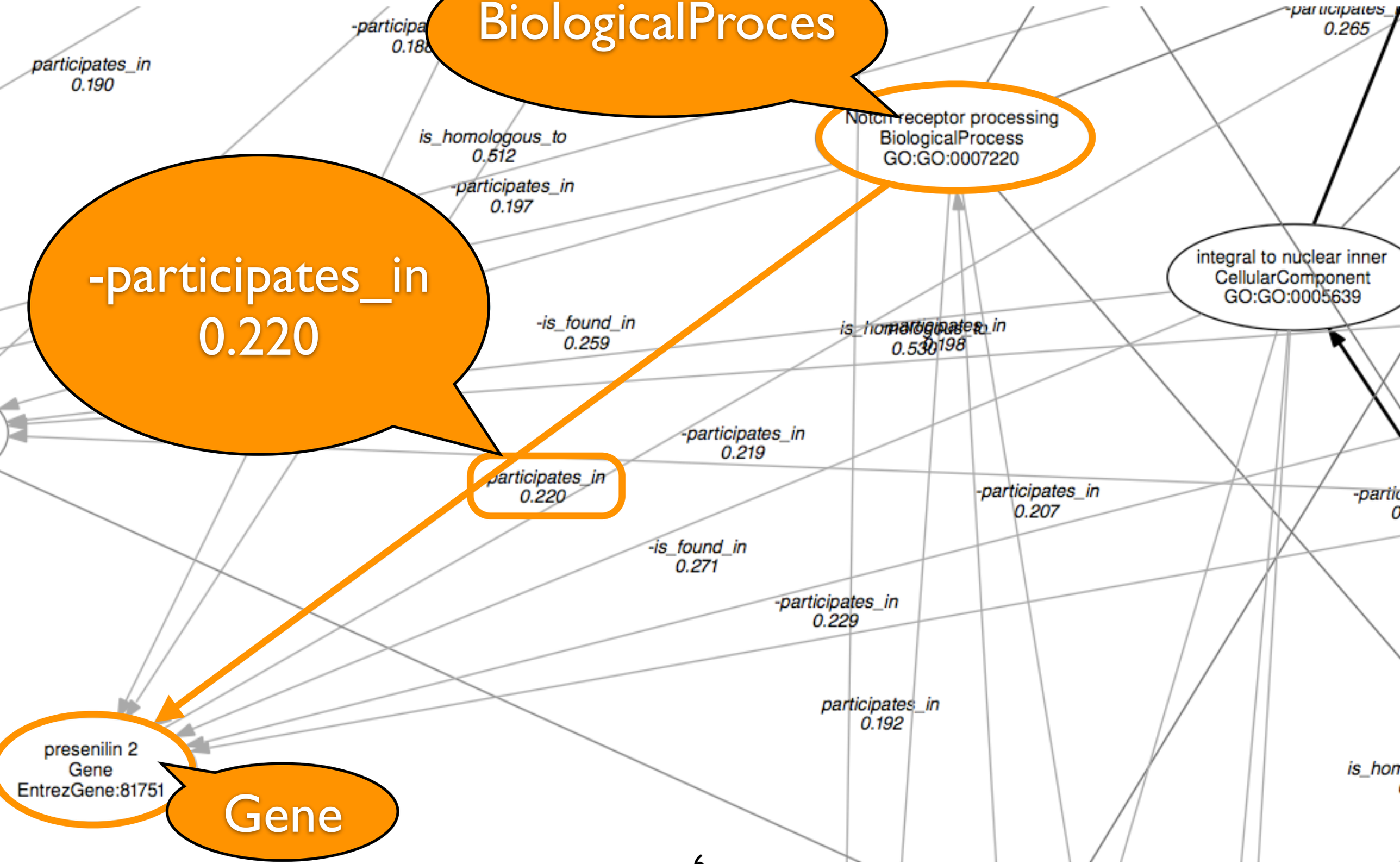


Notch receptor processing
BiologicalProcess
GO:GO:0007220

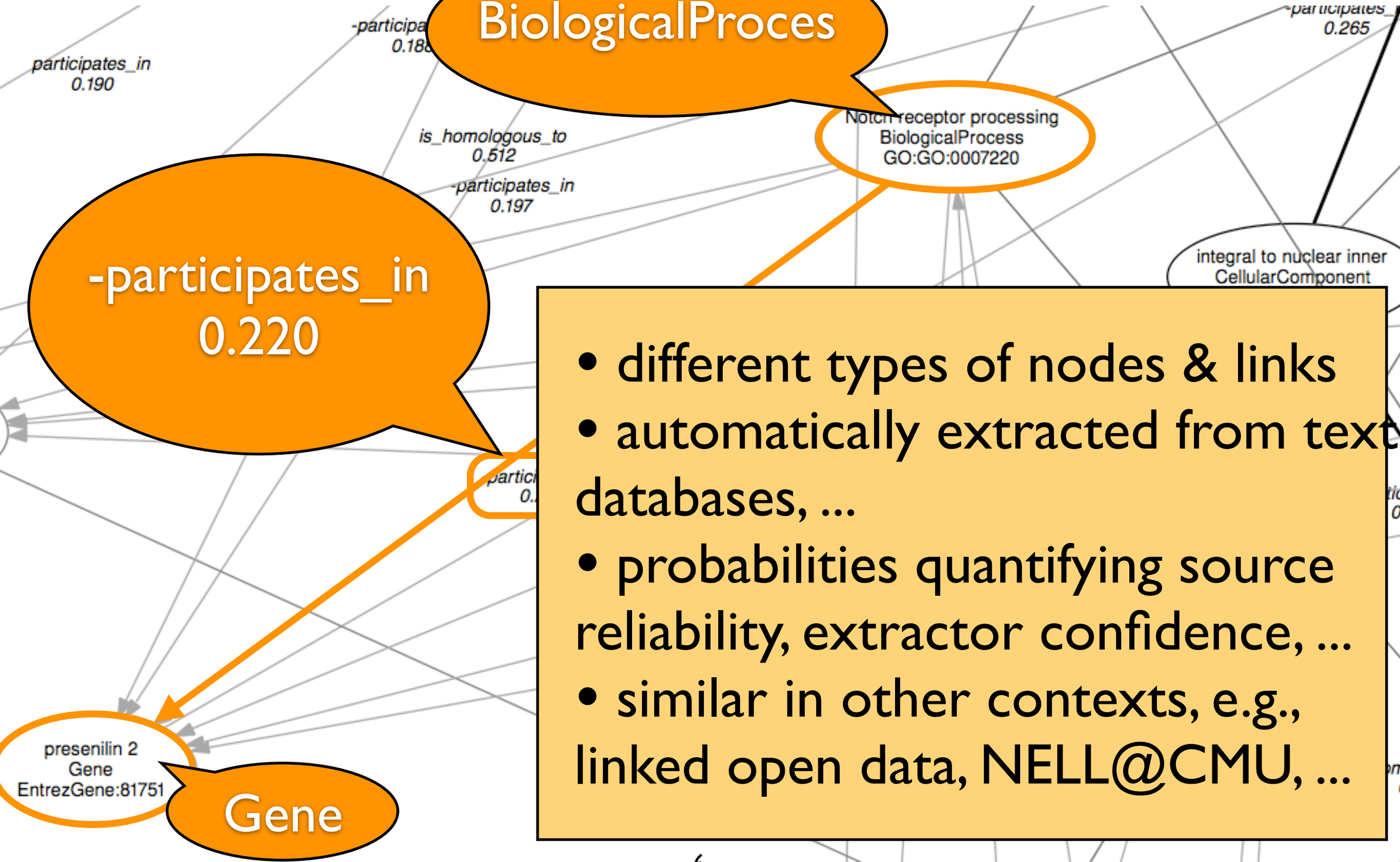
presenilin 2
Gene
EntrezGene:81751



Biomine Network



Biomine Network



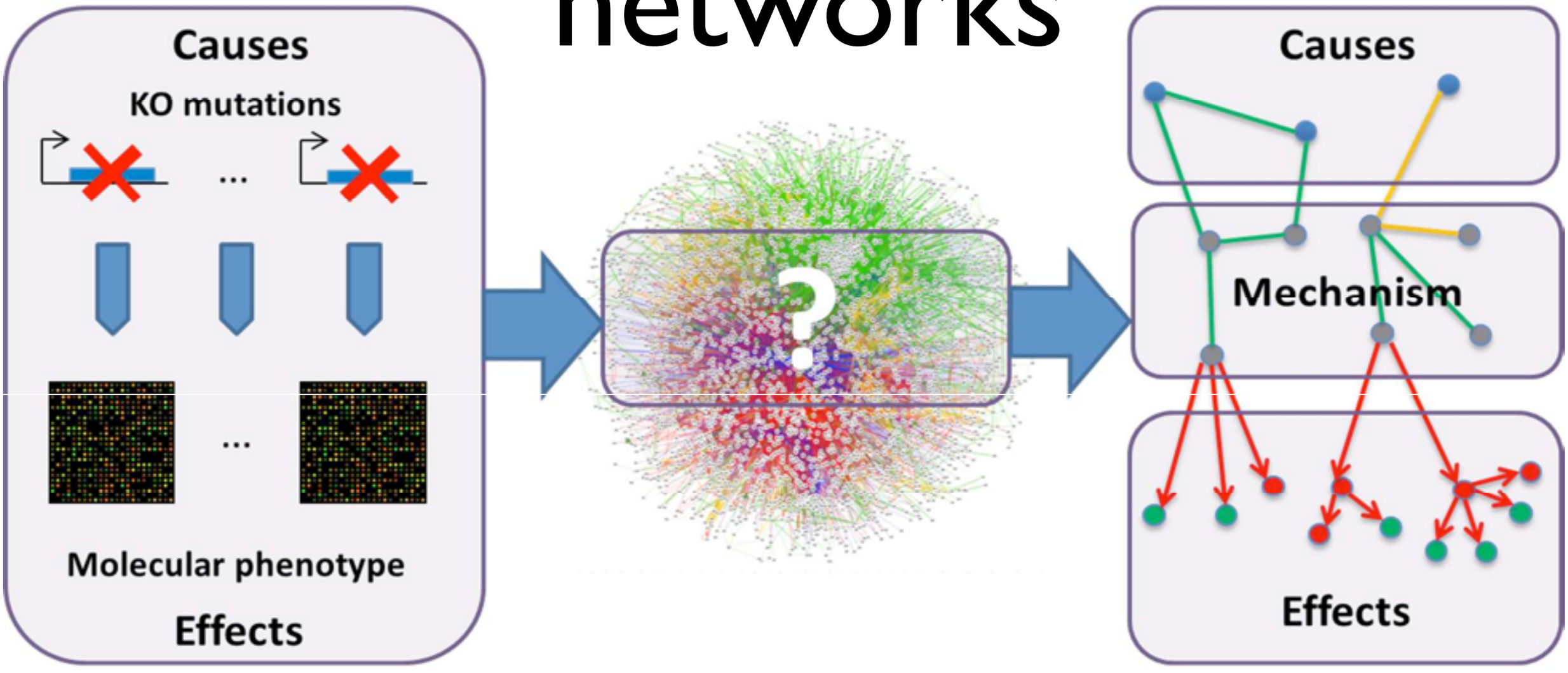
BiologicalProces

-participates_in
0.220

Gene

- different types of nodes & links
- automatically extracted from text databases, ...
- probabilities quantifying source reliability, extractor confidence, ...
- similar in other contexts, e.g., linked open data, NELL@CMU, ...

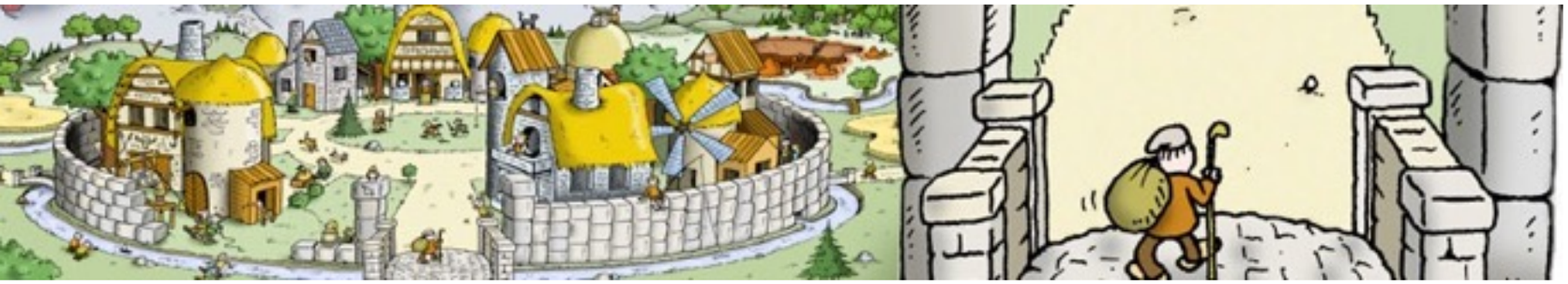
Molecular interaction networks



Can we find the mechanism connecting causes to effects?

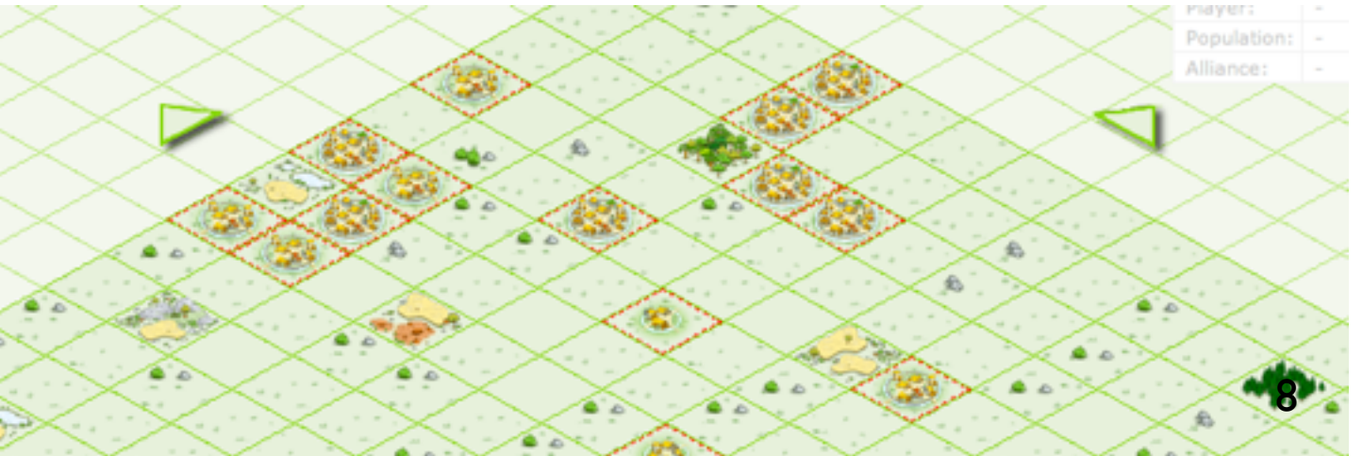
[De Maeyer et al, Molecular Biosystems 13]

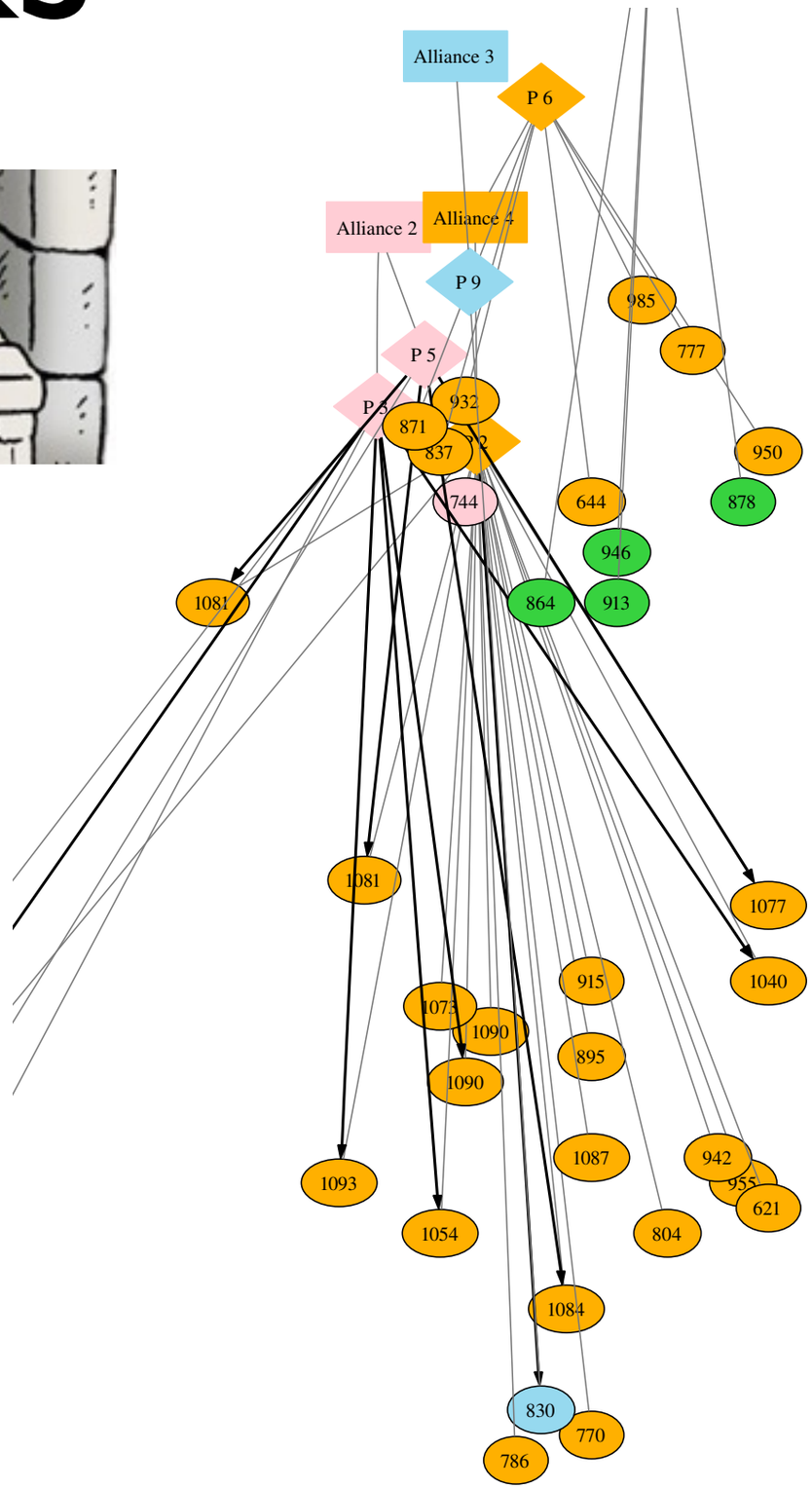
Dynamic networks



Travian: A massively multiplayer real-time strategy game

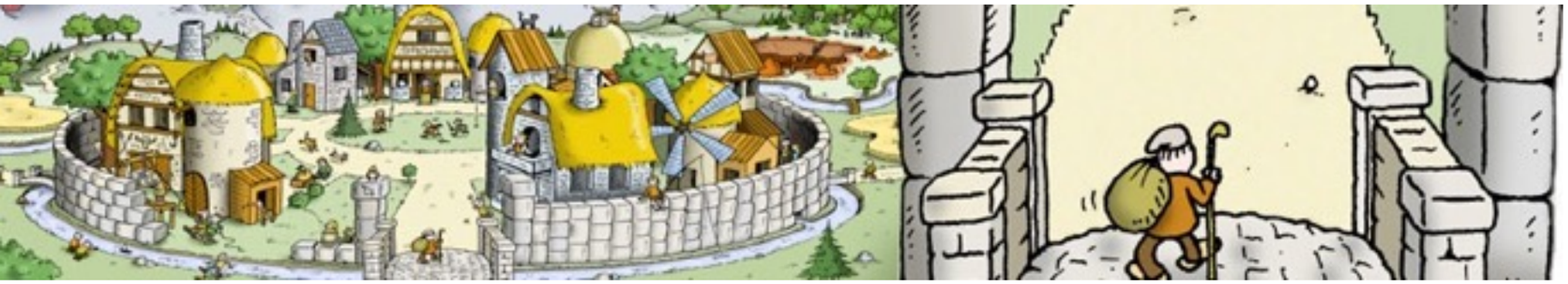
Can we build a model
of this world ?
Can we use it for playing
better ?





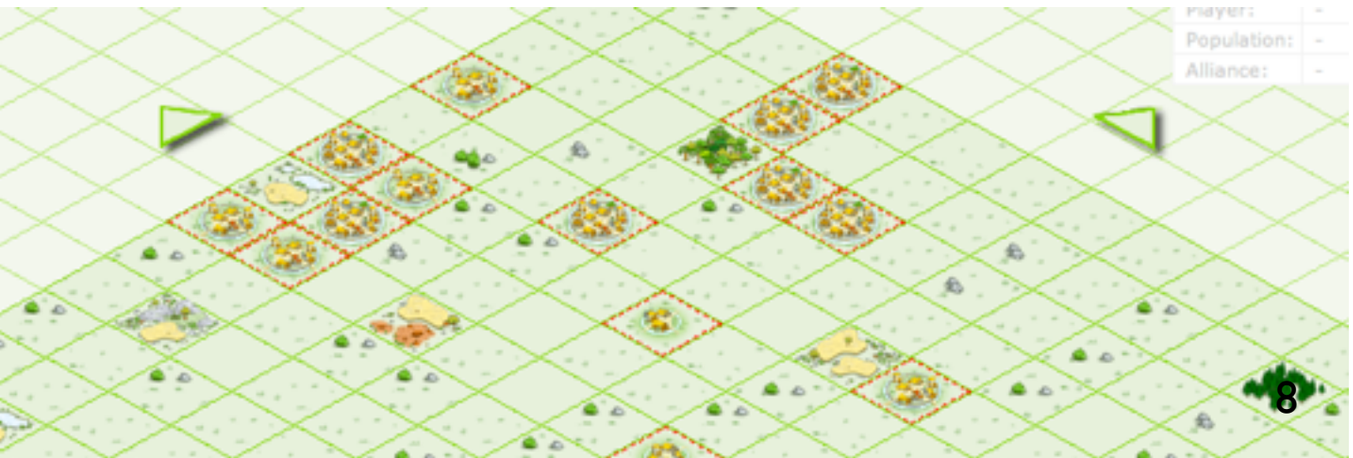
[Thon et al, MLJ 11]

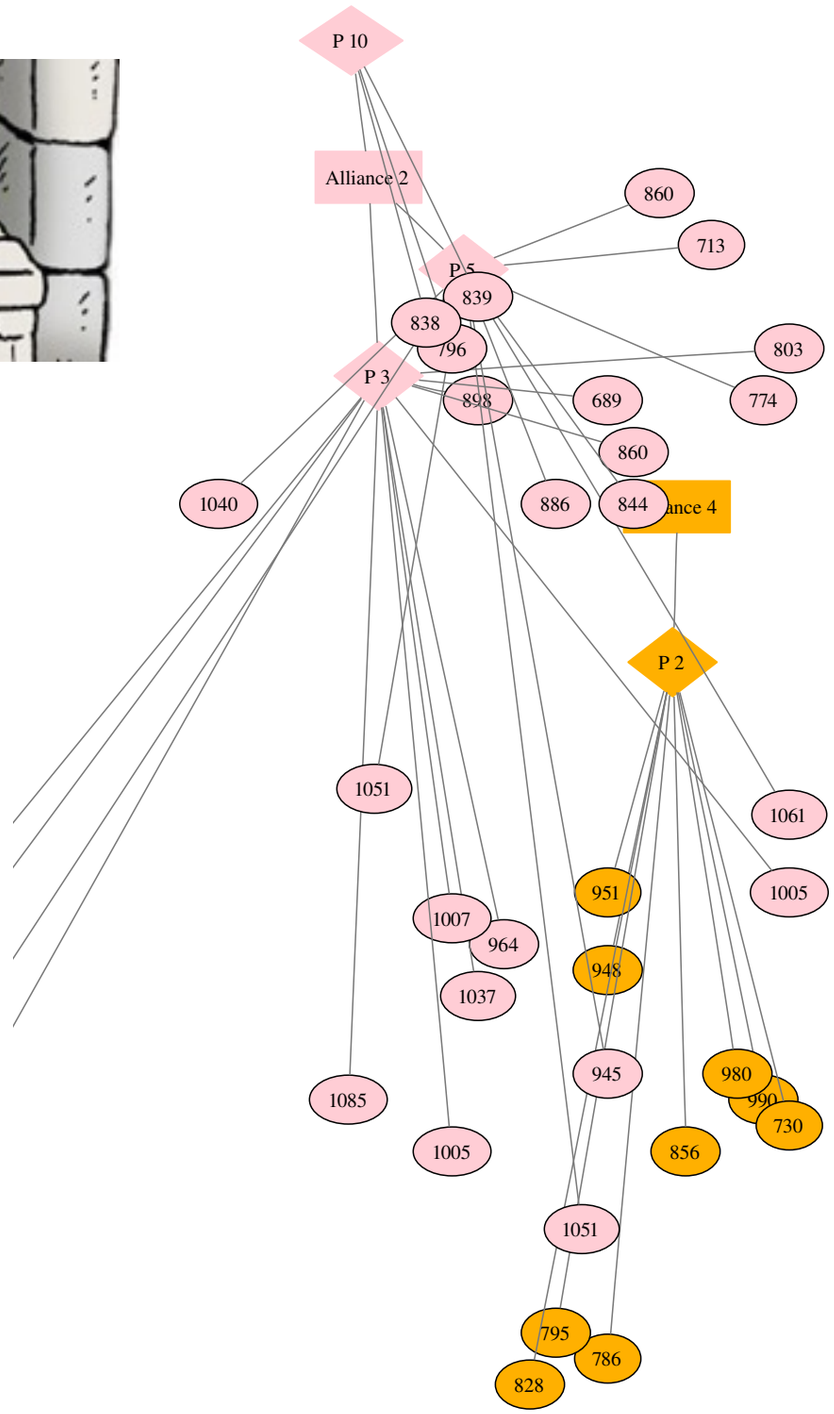
Dynamic networks



Travian: A massively multiplayer real-time strategy game

Can we build a model
of this world ?
Can we use it for playing
better ?

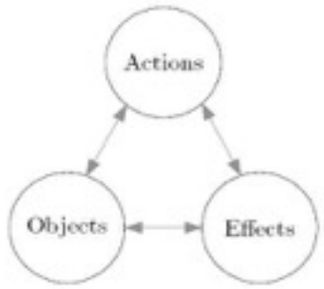




[Thon et al, MLJ 11]

Learning relational affordances

Learn probabilistic model



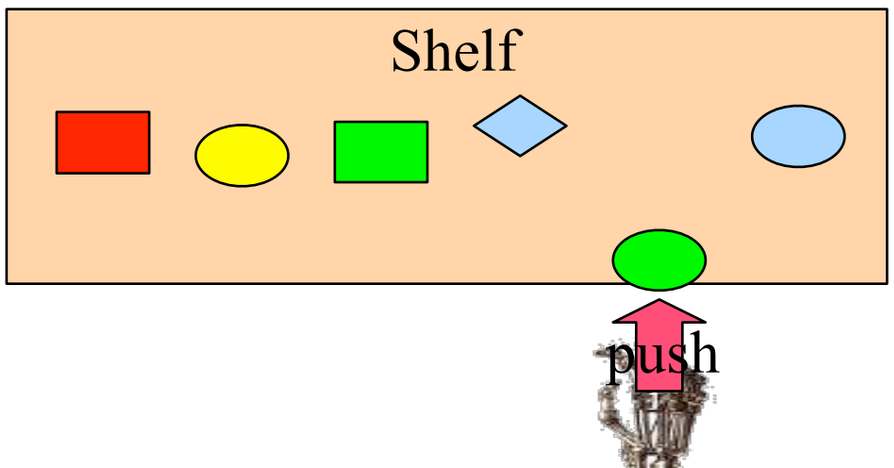
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

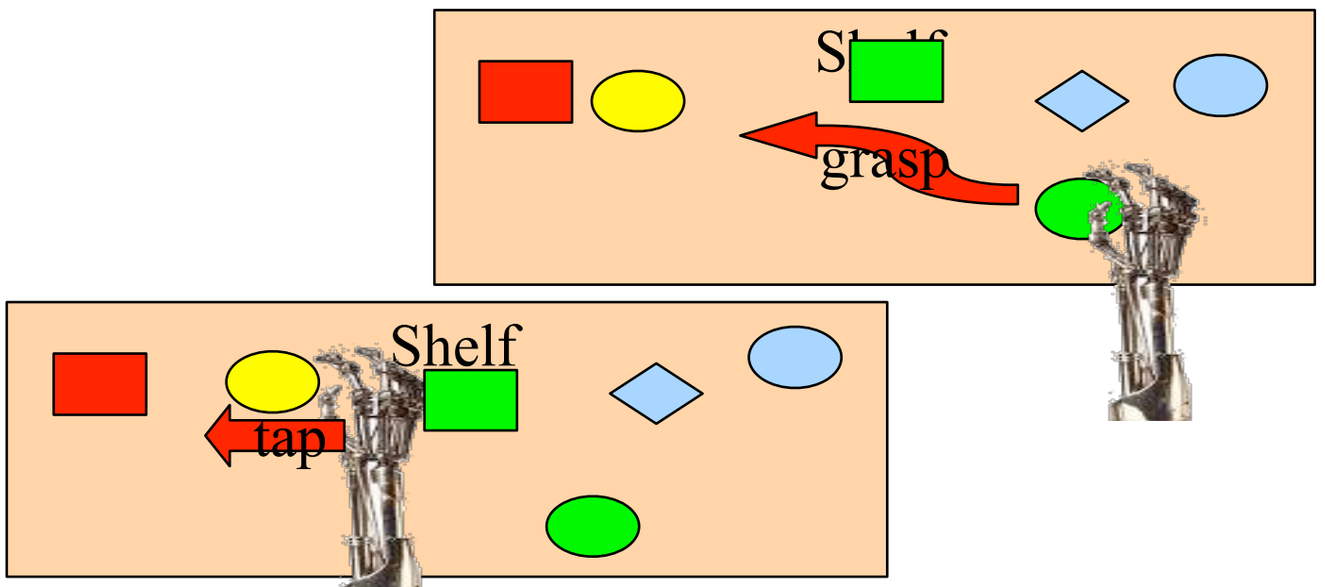
Learning relational affordances between two objects (learnt by experience)

From two object interactions

Generalize to N

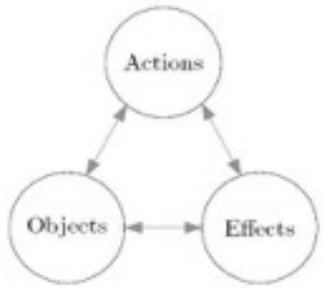
***Moldovan** et al. ICRA 12, 13, 14, PhD 15*





Learning relational affordances

Learn probabilistic model



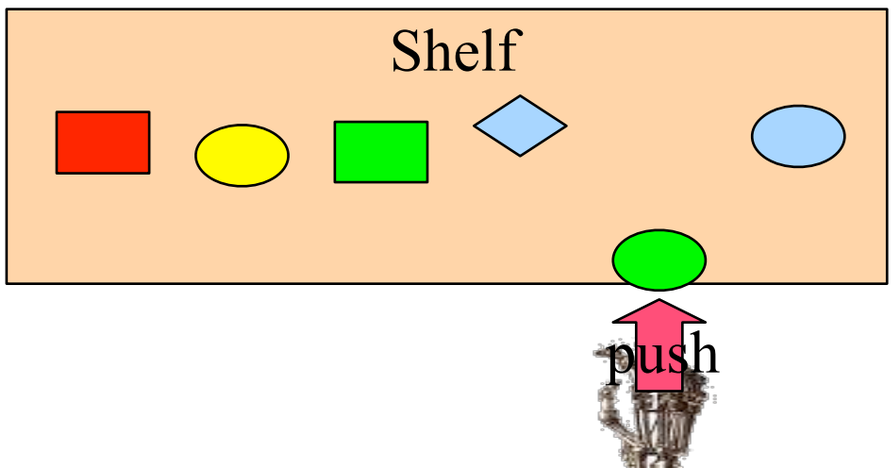
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

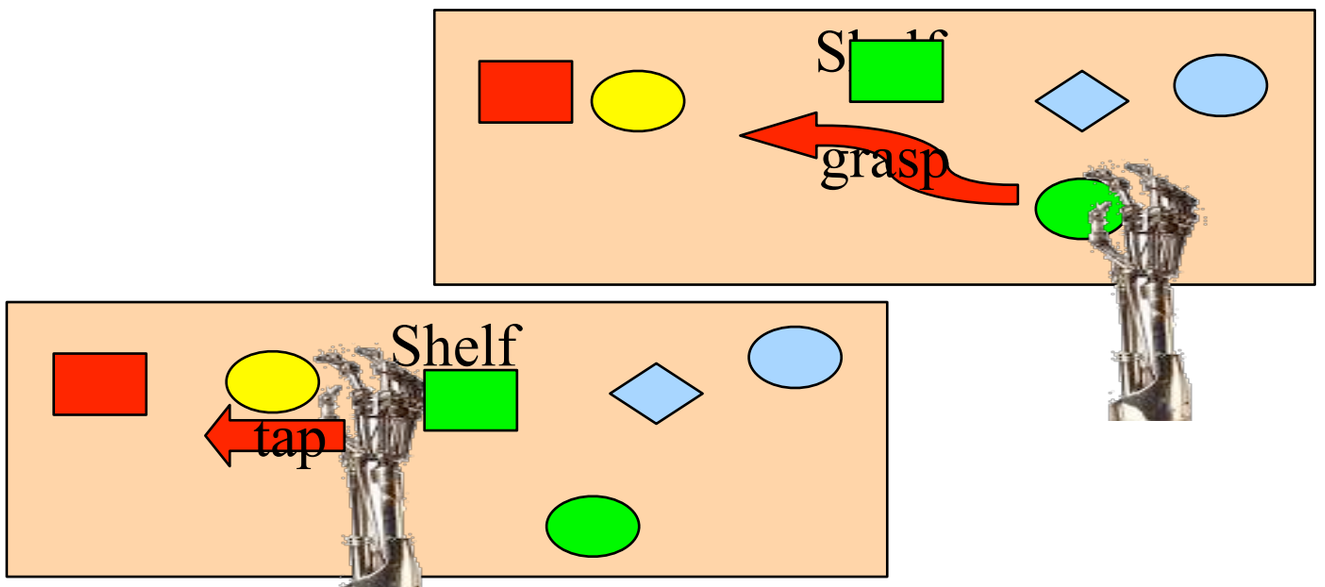
Learning relational affordances between two objects (learnt by experience)

From two object interactions

Generalize to N

***Moldovan** et al. ICRA 12, 13, 14, PhD 15*





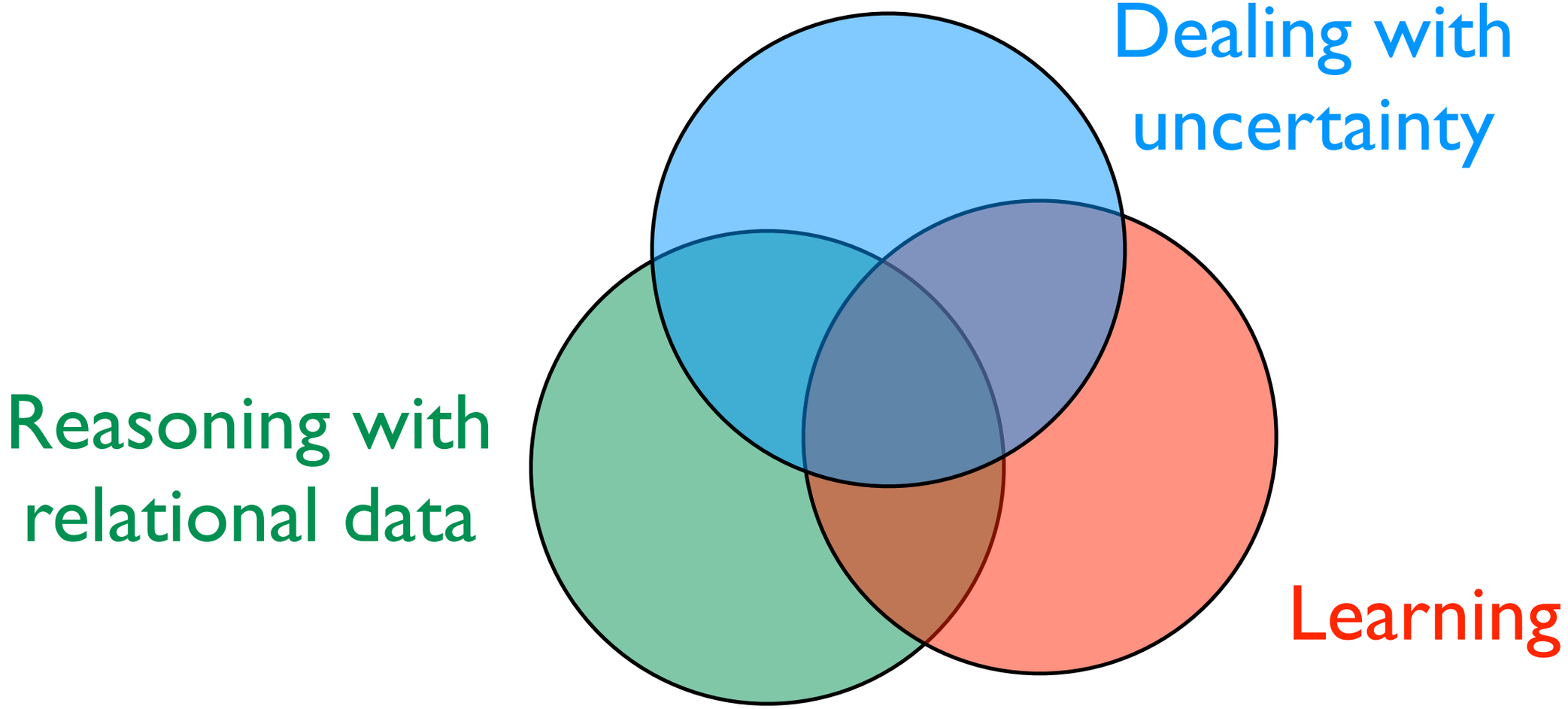
Answering Probability Questions

Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

The answer is 0.235941.

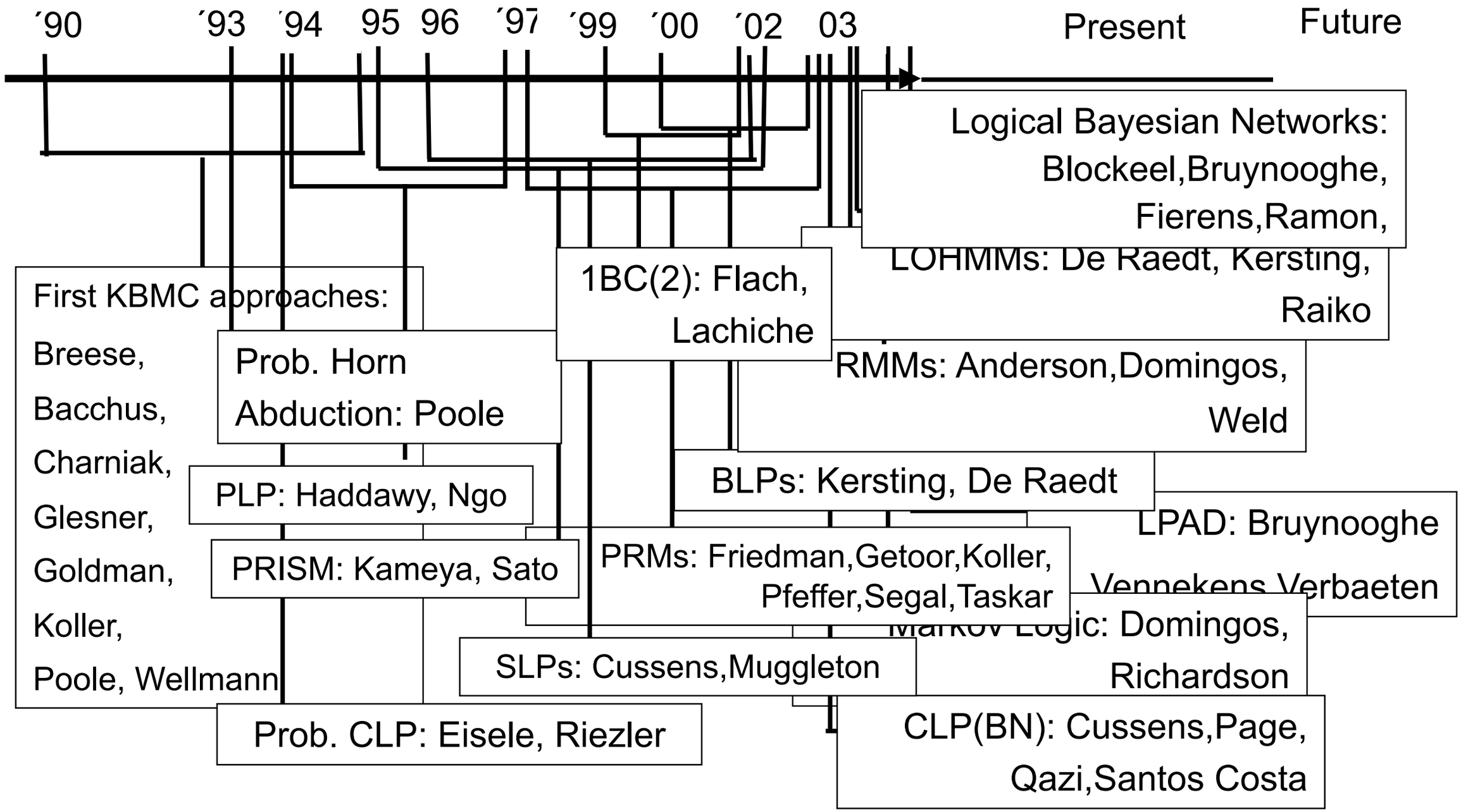
[Dries et al., IJCAI 17]

Common theme



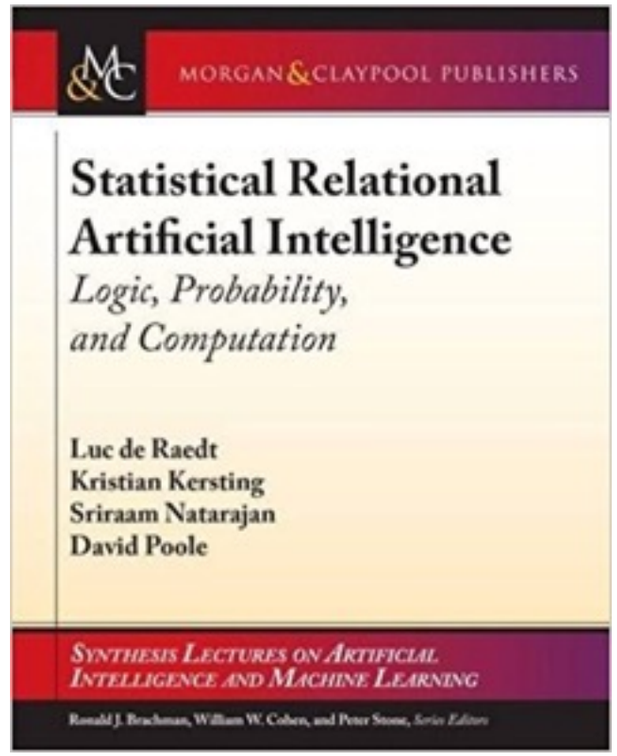
Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Some formalisms



Many different angles

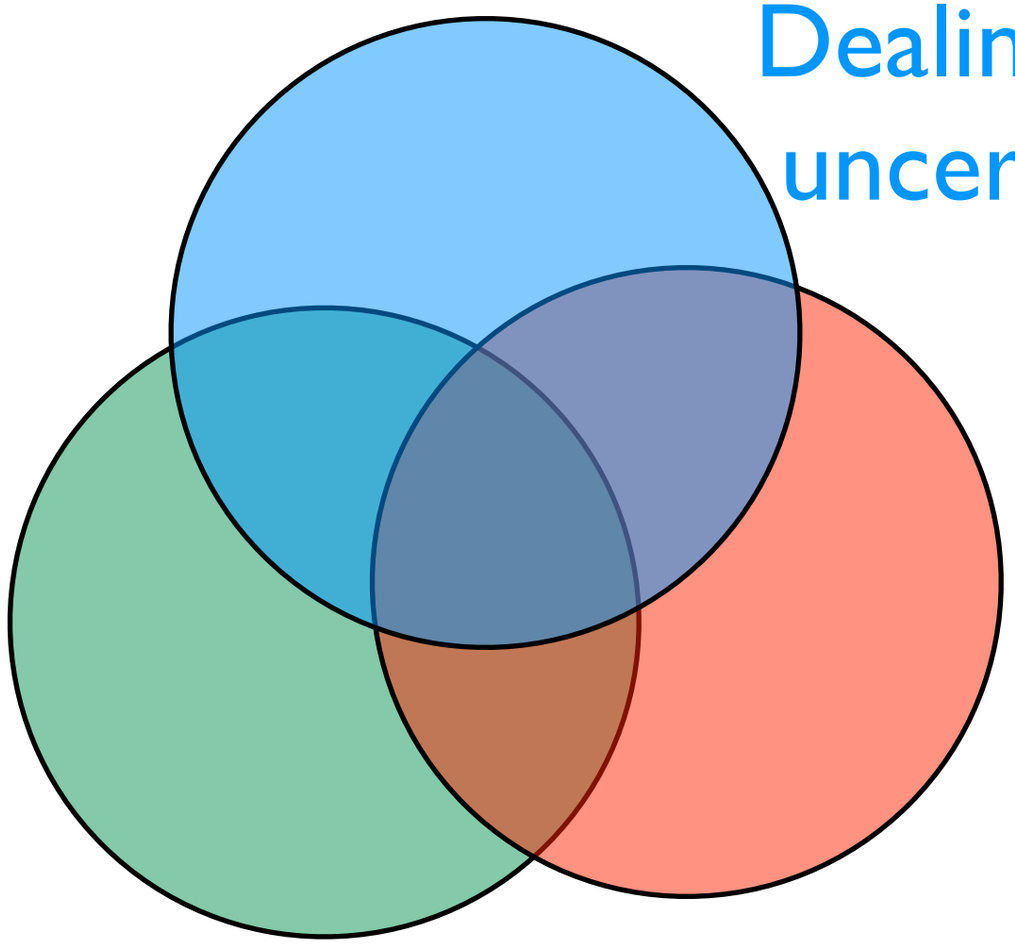
- Probabilistic programming
  - Logic programming and probabilistic databases (ProbLog and DS as representatives)
  - Functional and imperative (Church as representatives)
- Statistical relational AI and learning
  - Markov Logic



ProbLog

probabilistic Prolog

Dealing with uncertainty



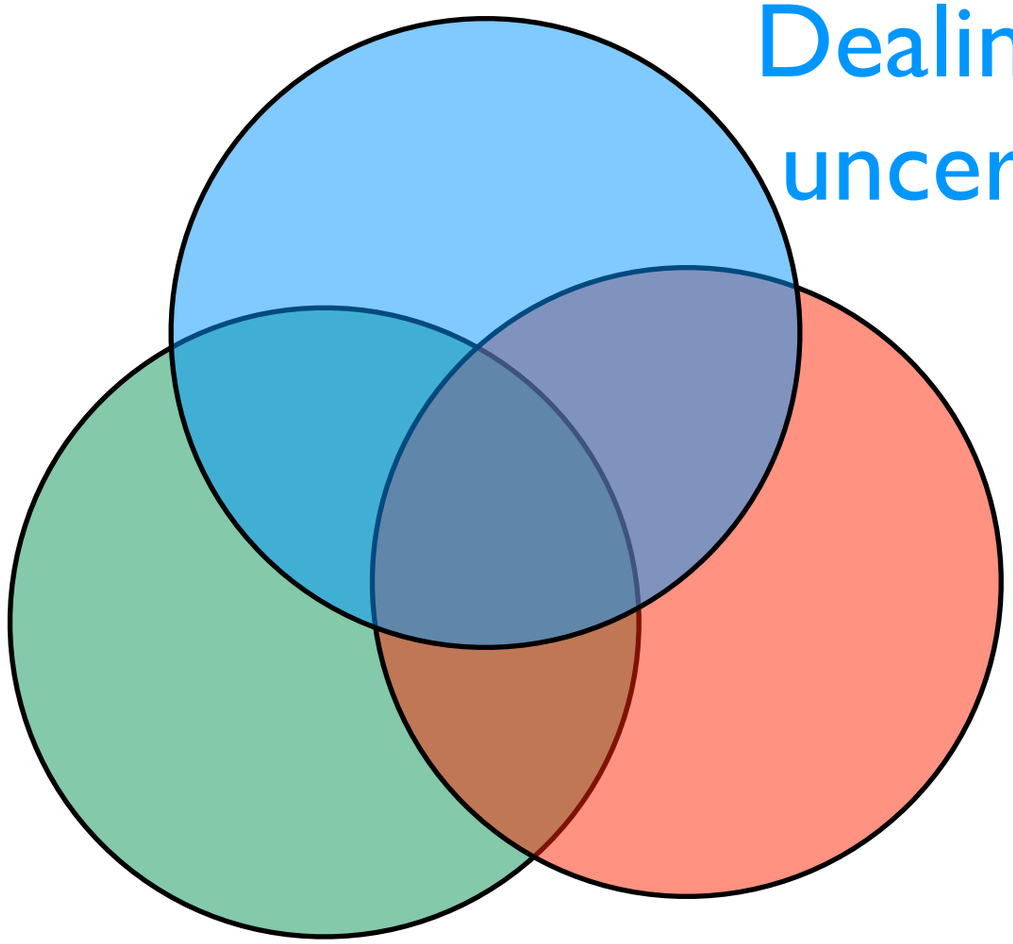
Reasoning with relational data

Learning

http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog

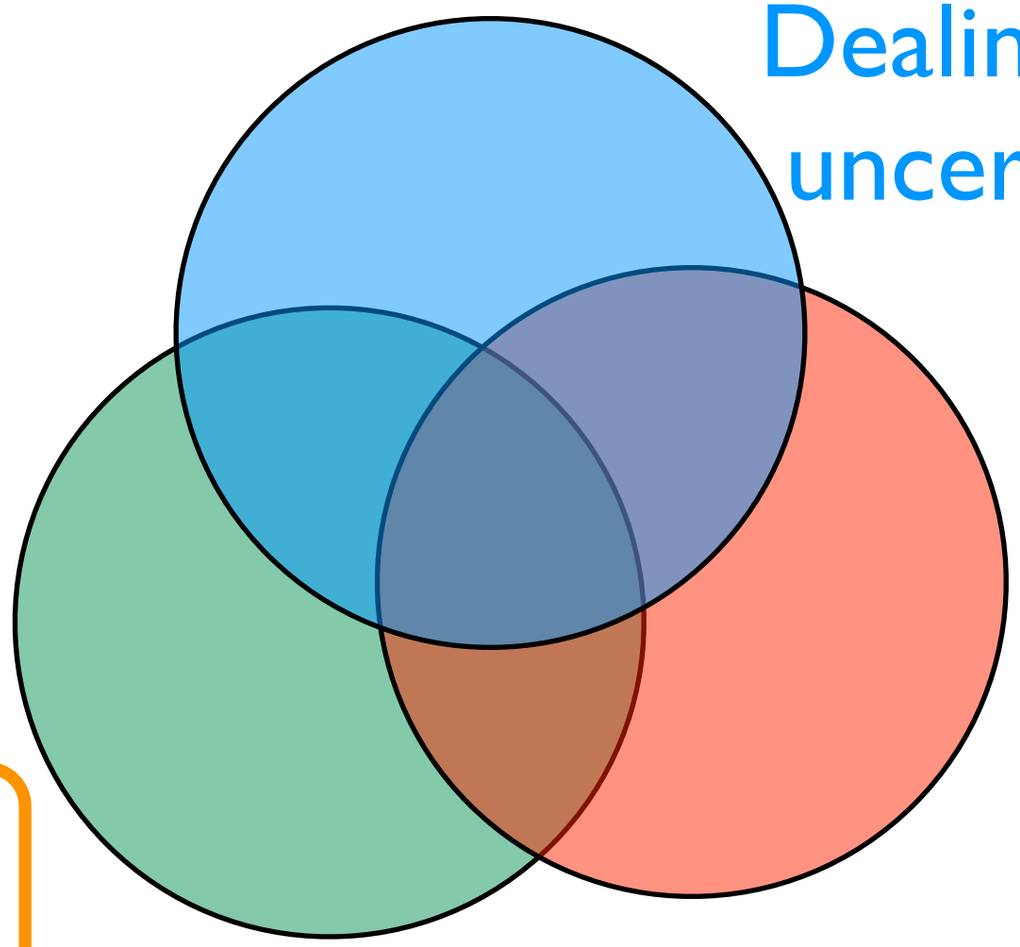



```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog




```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

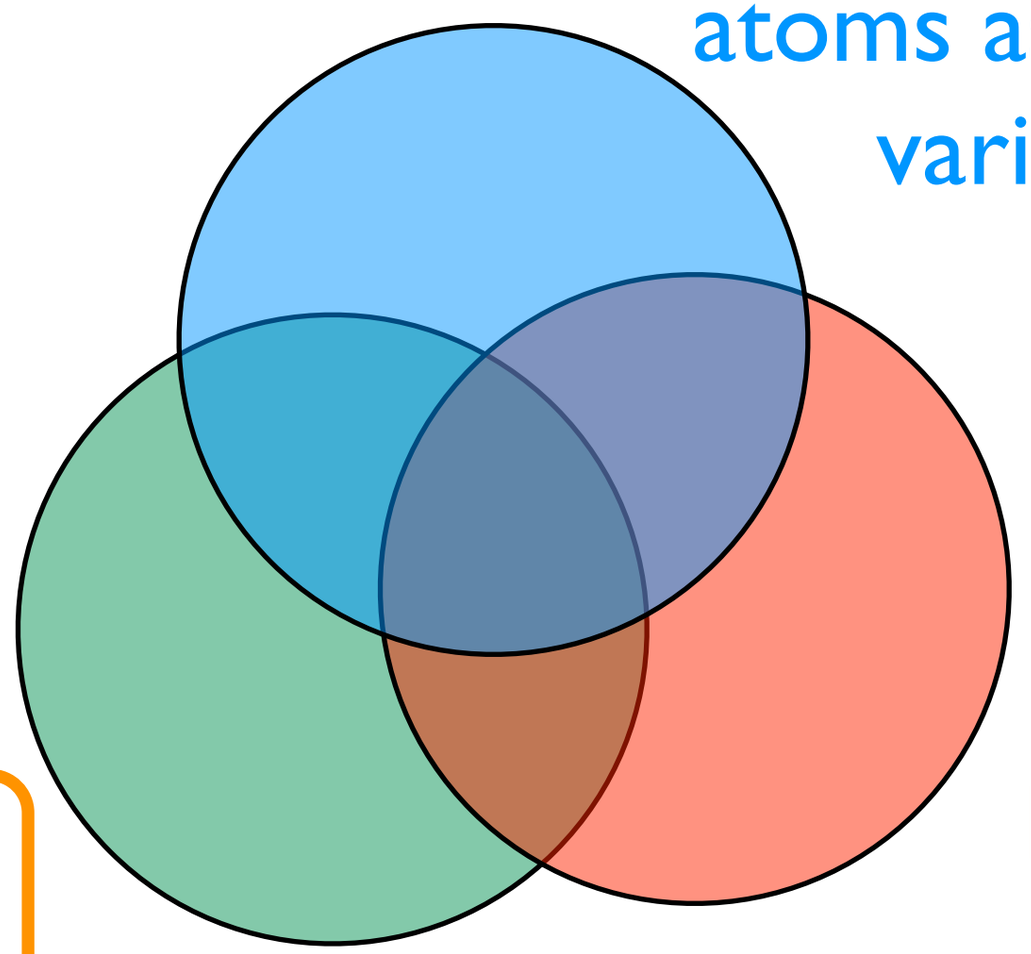

http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

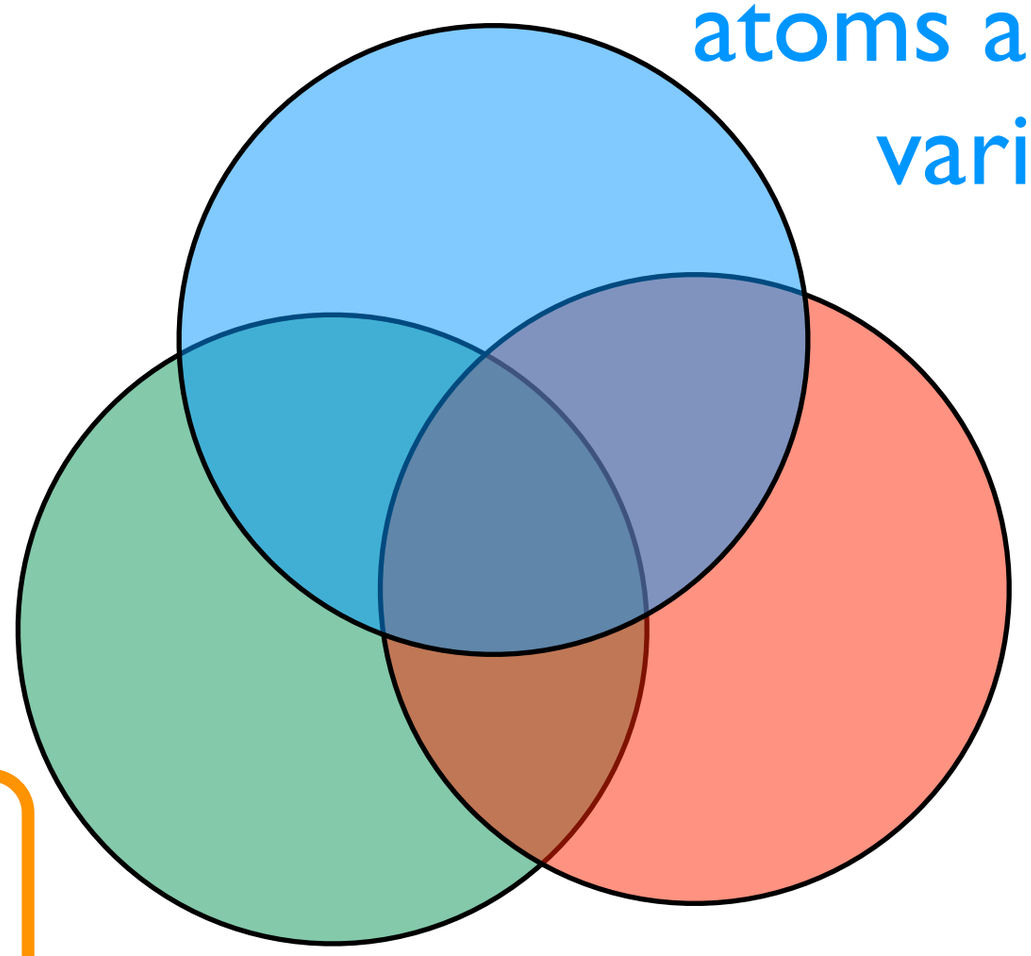
ProbLog

probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

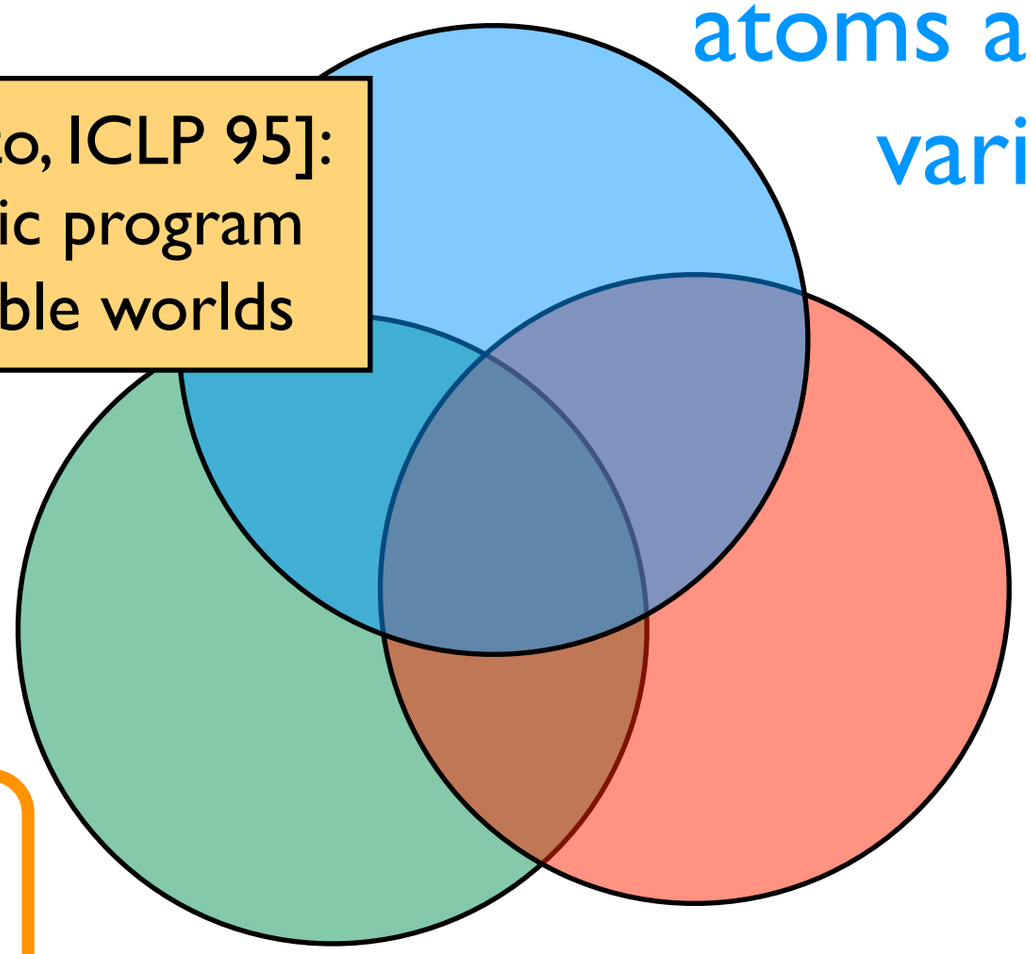
probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

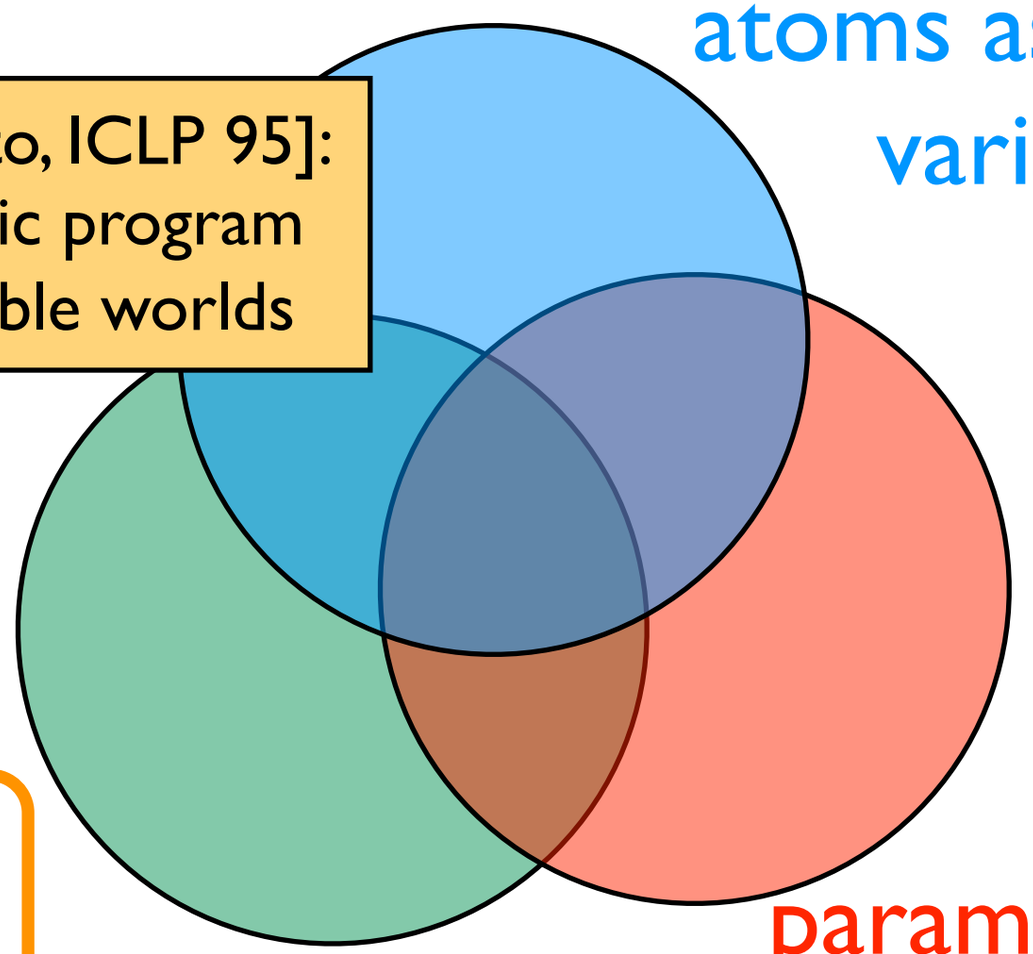
probabilistic Prolog

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

parameter learning, adapted relational learning techniques

http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds

e.g., PRISM, ICL, ProbLog, LPADs, CP-logic, PITA, ...

multi-valued switches

probabilistic alternatives

probabilistic facts

annotated disjunctions

causal-probabilistic laws

Roadmap

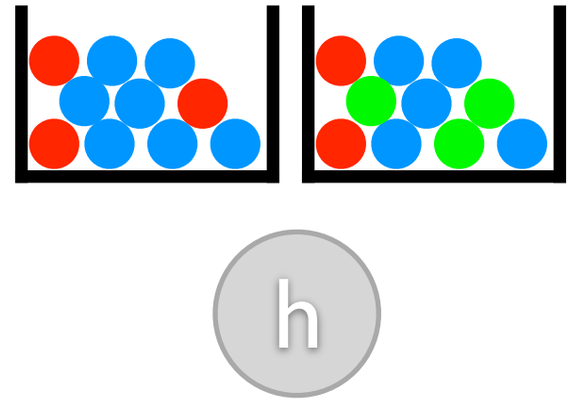
1. Modeling
2. Inference
3. Learning
4. Applications

... with some detours on the way

Part 1 : Modeling

ProbLog by example:

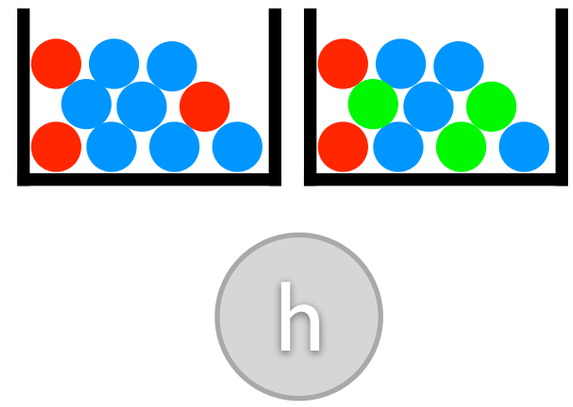
A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

ProbLog by example:

A bit of gambling



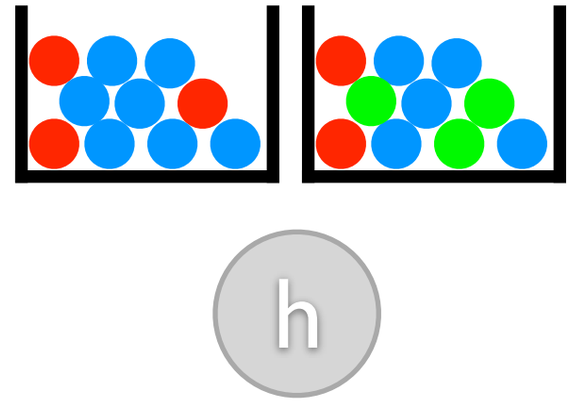
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.
```

probabilistic fact: heads is true with probability 0.4 (and false with 0.6)

ProbLog by example:

A bit of gambling



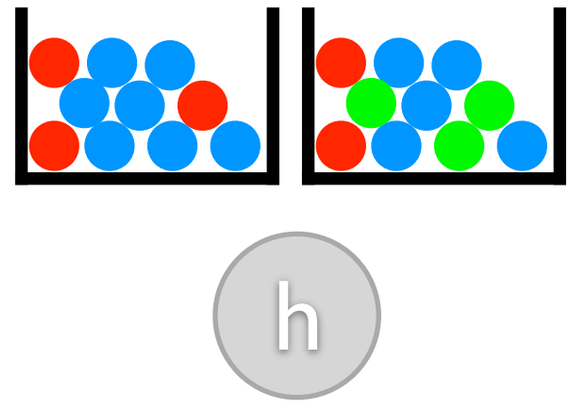
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
```

annotated disjunction: first ball is red with probability 0.3 and blue with 0.7

ProbLog by example:

A bit of gambling



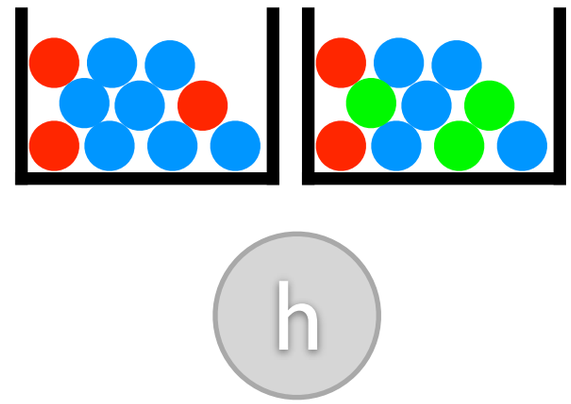
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

annotated disjunction: second ball is red with probability 0.2, green with 0.3, and blue with 0.5

ProbLog by example:

A bit of gambling



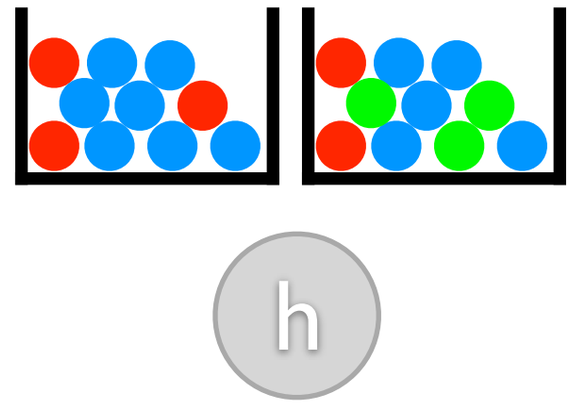
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



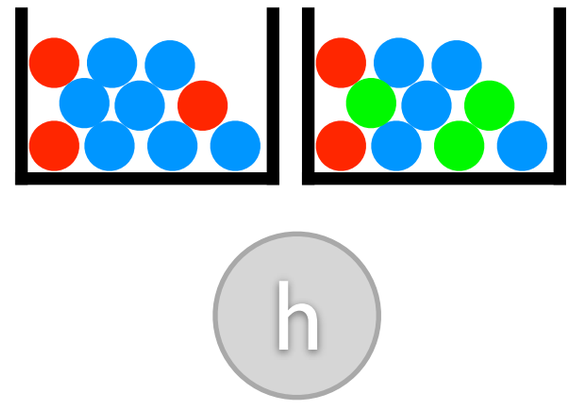
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

probabilistic choices

```
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

consequences

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

- Probability of **`win`**?
- Probability of **`win`** given **`col(2,green)`**?
- Most probable world where **`win`** is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of `win`?

query

- Probability of `win` given `col(2,green)`?
- Most probable world where `win` is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given **`col(2,green)`**?

evidence

- Most probable world where **`win`** is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given **`col(2,green)`**?

- Most probable world where **`win`** is true?

MPE inference

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

Possible Worlds

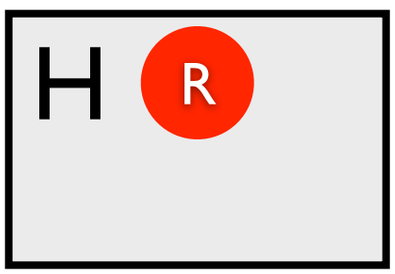
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

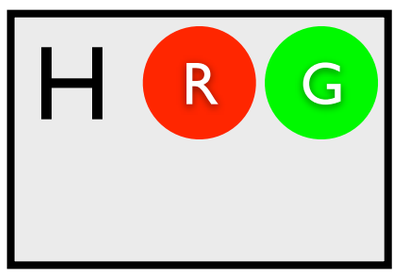
0.4 ×0.3



Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

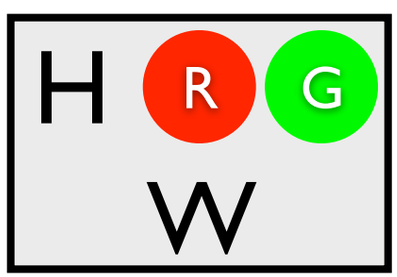
0.4 ×0.3 ×0.3



Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



Possible Worlds

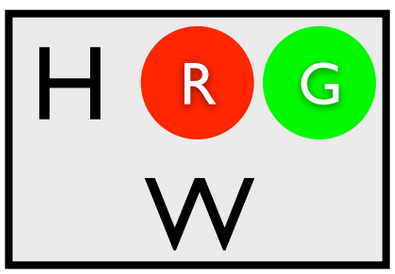
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$ $(1-0.4)$

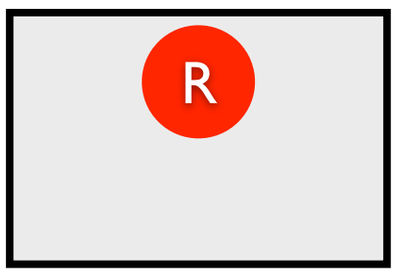
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



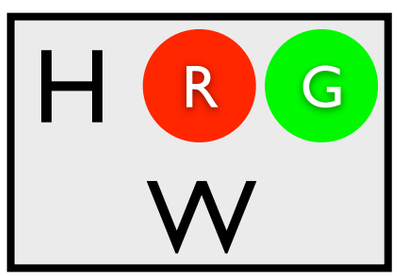
(1−0.4)×0.3



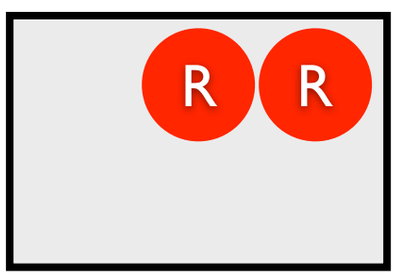
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$



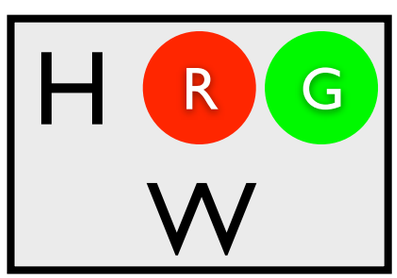
$(1-0.4) \times 0.3 \times 0.2$



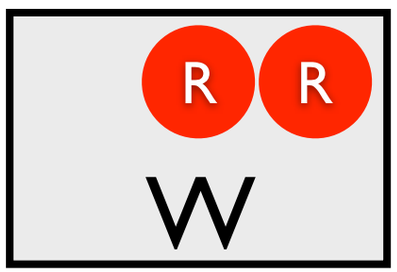
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$



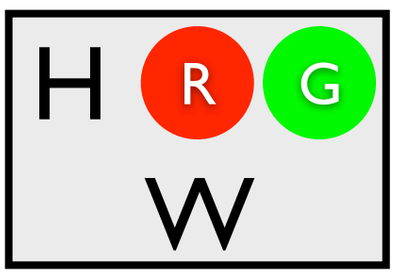
$(1-0.4) \times 0.3 \times 0.2$



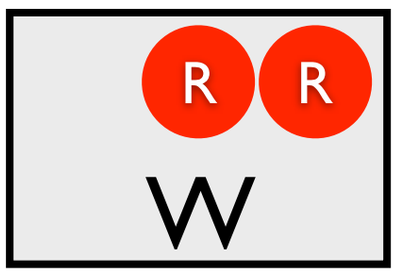
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



(1−0.4)×0.3 ×0.2

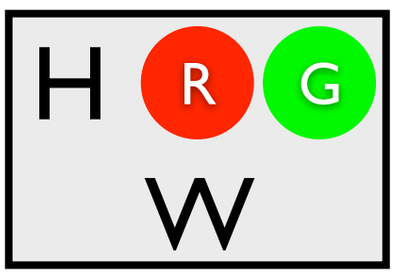


(1−0.4)

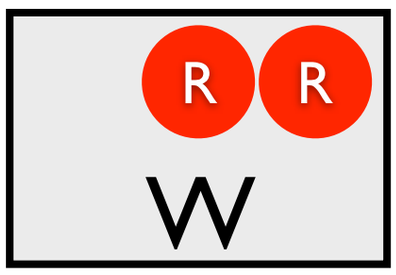
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

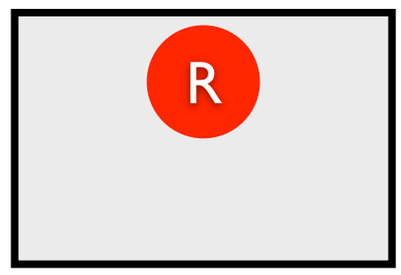
0.4 ×0.3 ×0.3



(1−0.4)×0.3 ×0.2



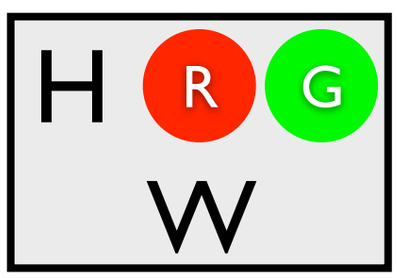
(1−0.4)×0.3



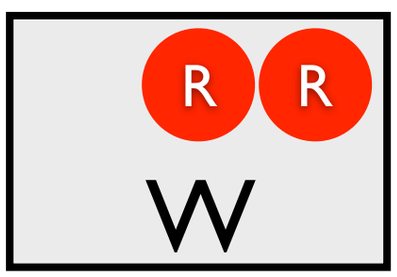
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

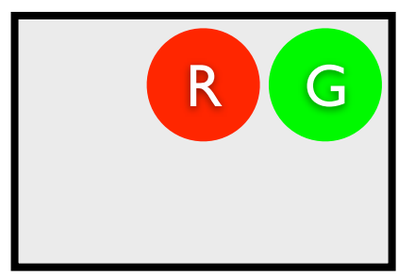
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



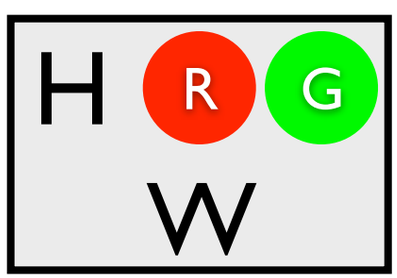
$(1-0.4) \times 0.3 \times 0.3$



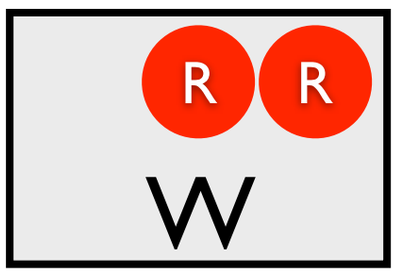
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

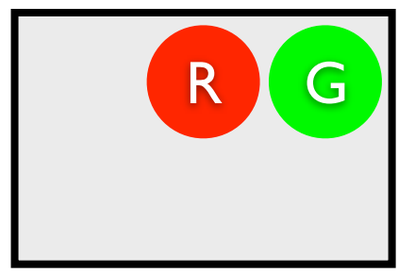
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



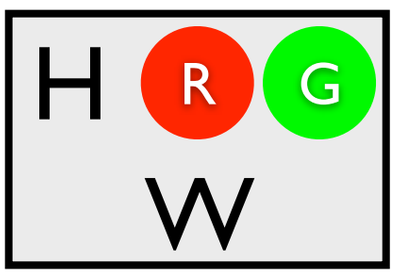
$(1-0.4) \times 0.3 \times 0.3$



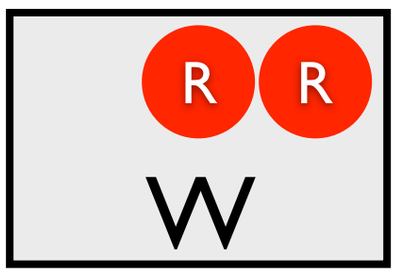
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

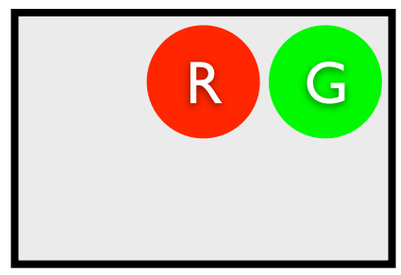
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$

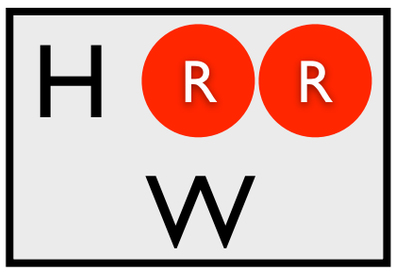


$(1-0.4) \times 0.3 \times 0.3$

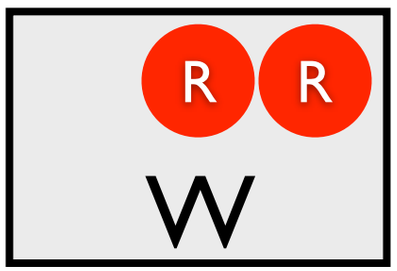


All Possible Worlds

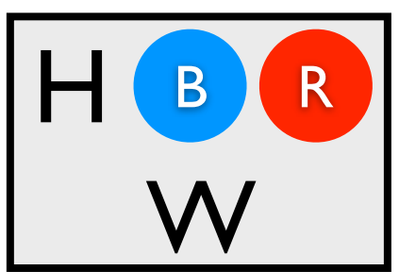
0.024



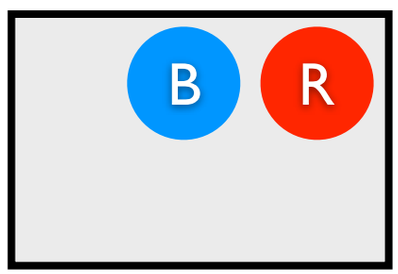
0.036



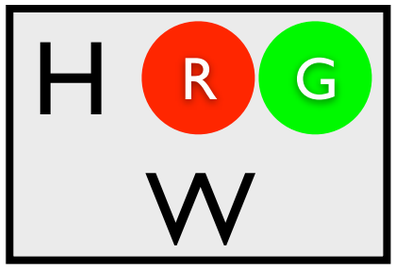
0.056



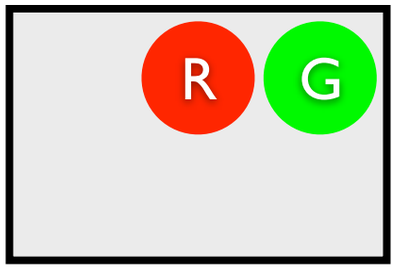
0.084



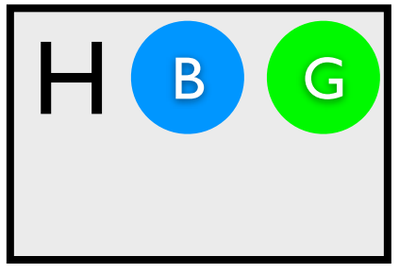
0.036



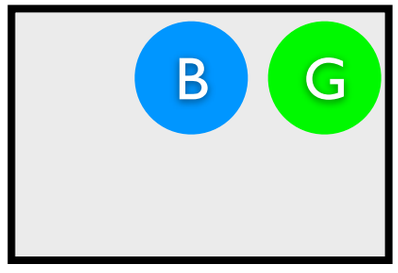
0.054



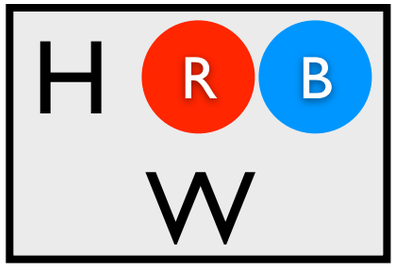
0.084



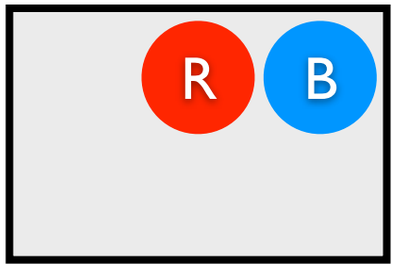
0.126



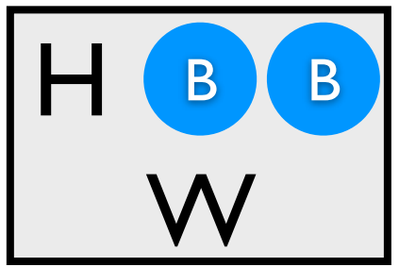
0.060



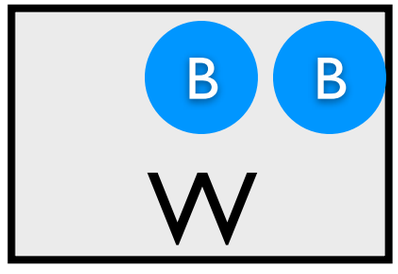
0.090



0.140

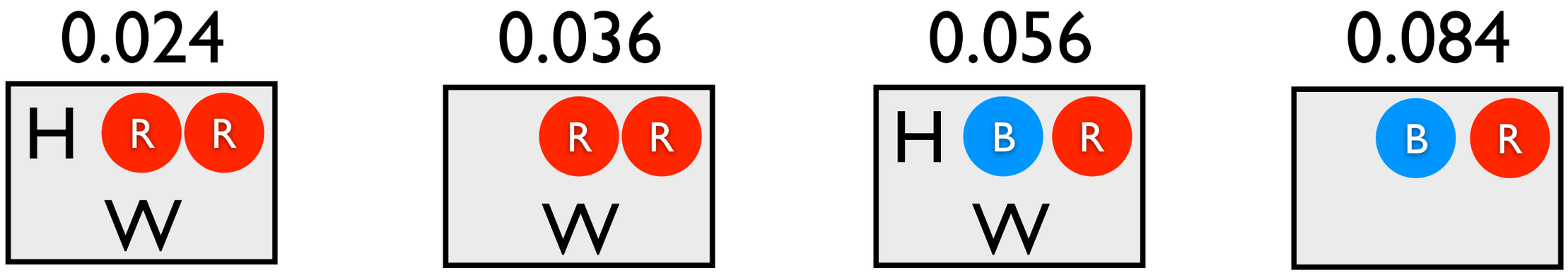


0.210



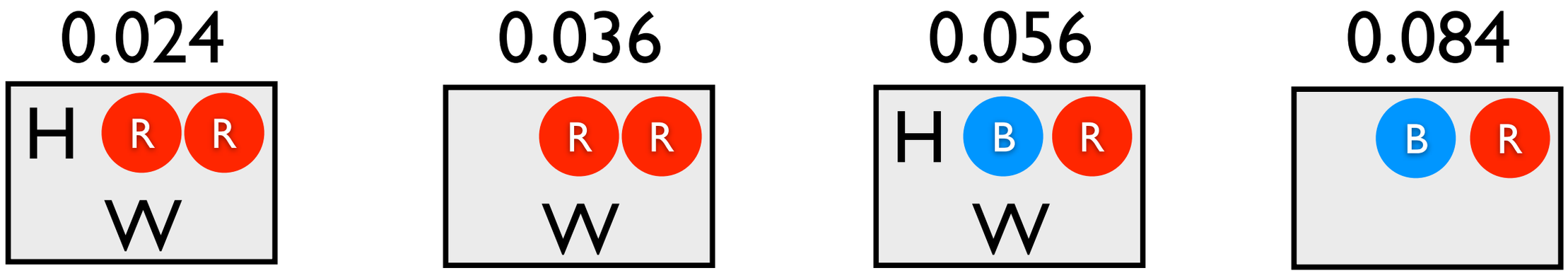
Most likely world where **win** is true?

MPE Inference



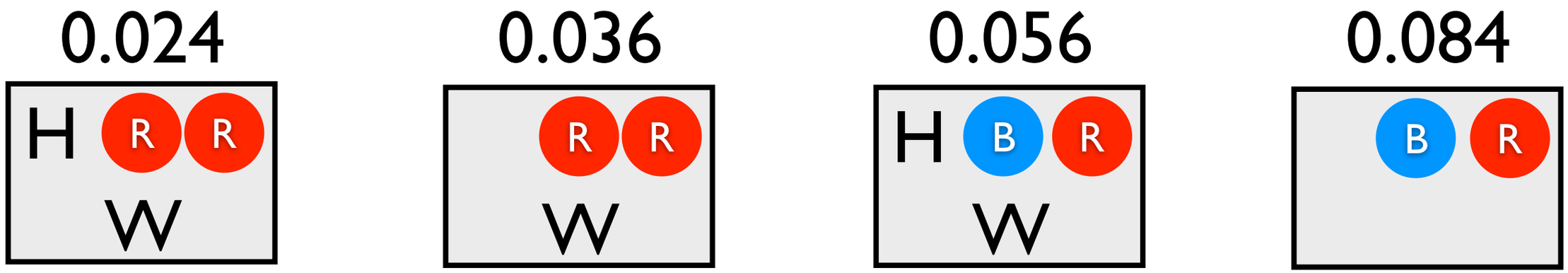
Most likely world where **win** is true?

MPE Inference



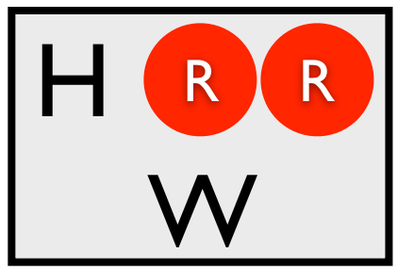
Most likely world where `col(2,blue)` is false?

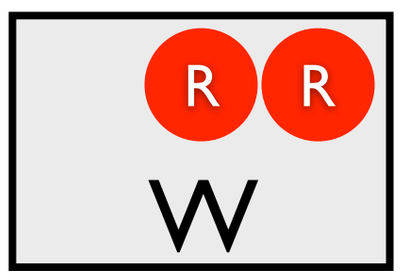
MPE Inference

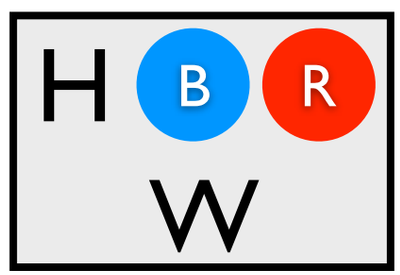


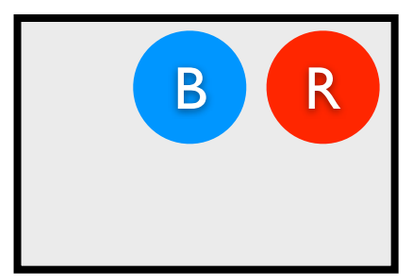
Most likely world where `col(2,blue)` is false?

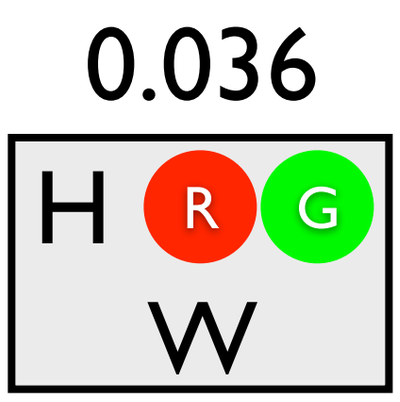
MPE Inference

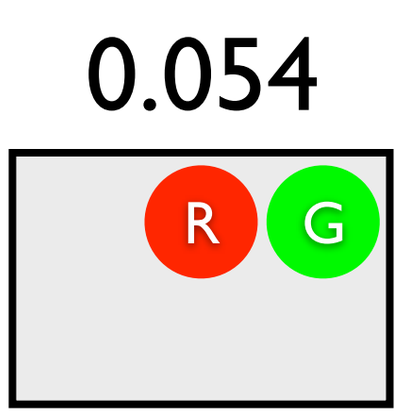


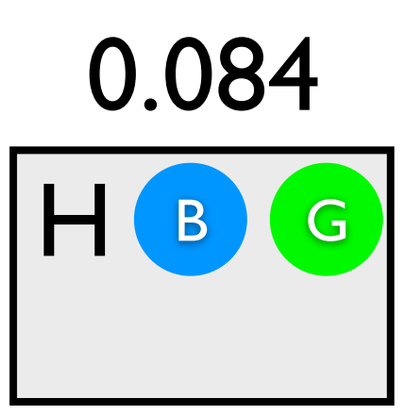


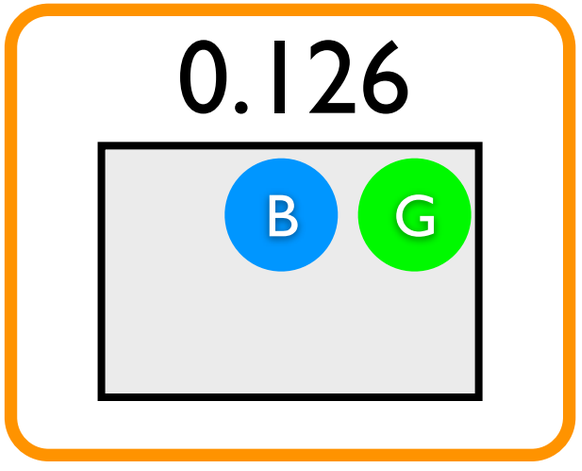


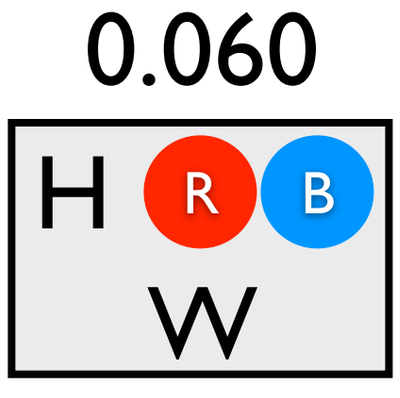


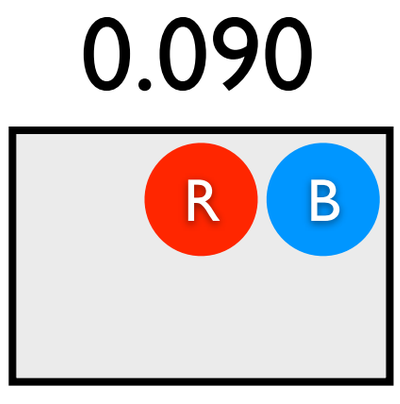


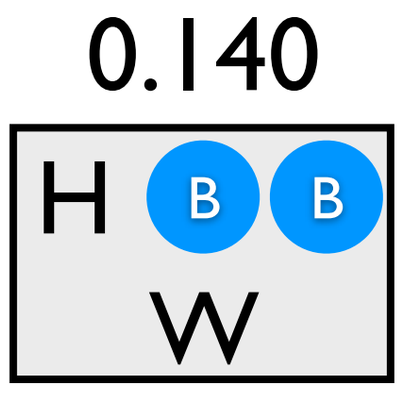


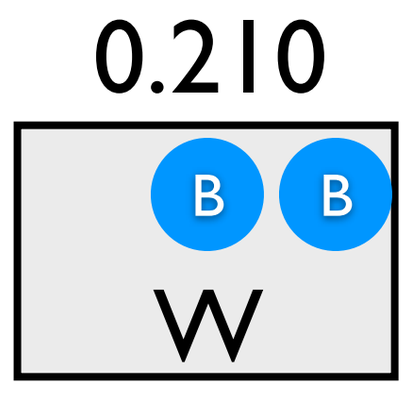








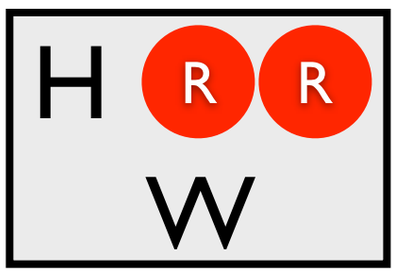




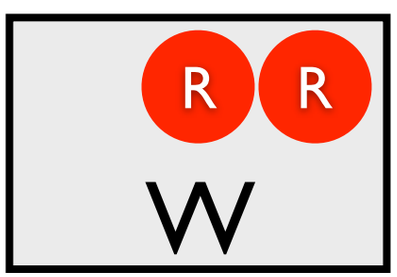
P(win)= ?

Marginal Probability

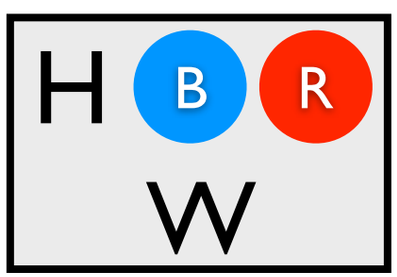
0.024



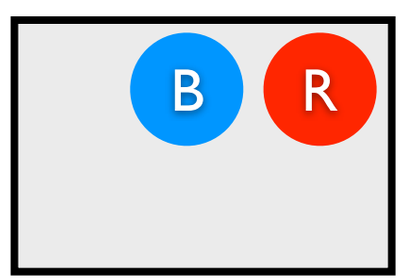
0.036



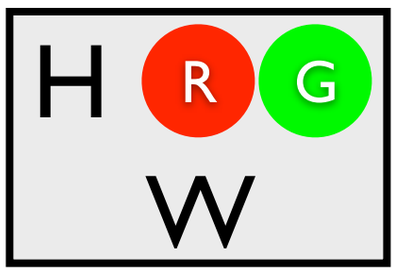
0.056



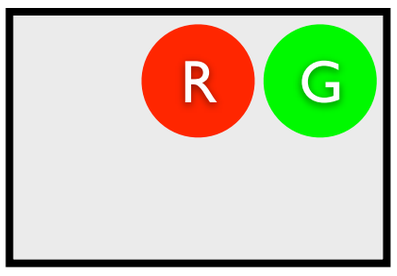
0.084



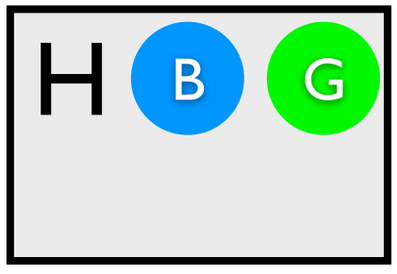
0.036



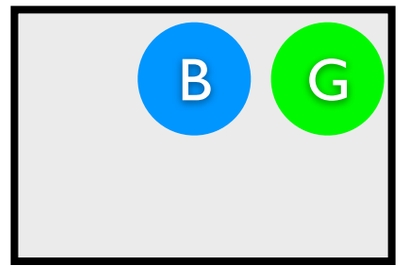
0.054



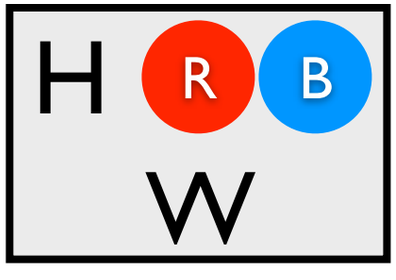
0.084



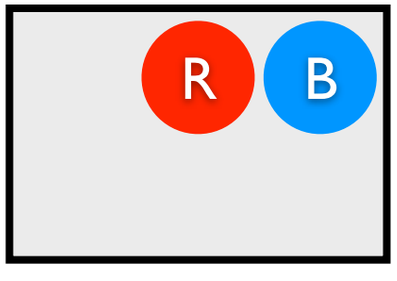
0.126



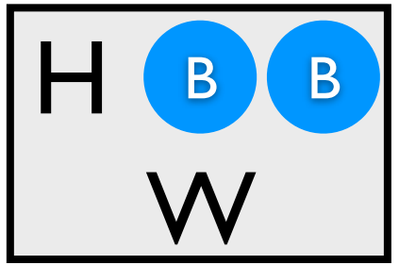
0.060



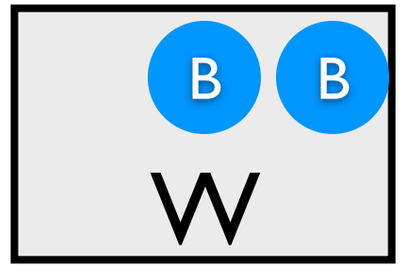
0.090



0.140



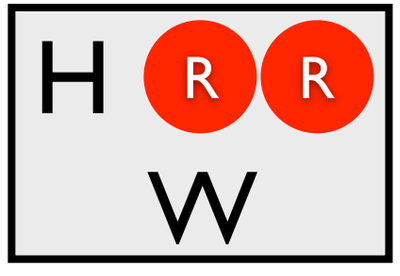
0.210



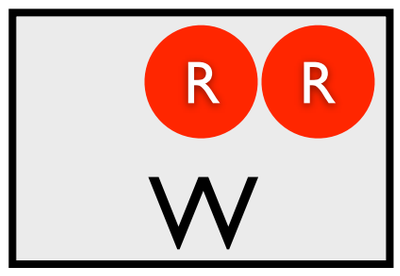
P(win)= Σ

Marginal Probability

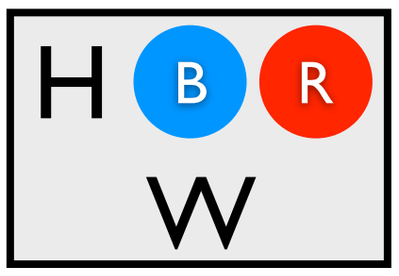
0.024



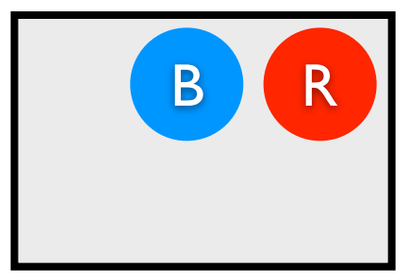
0.036



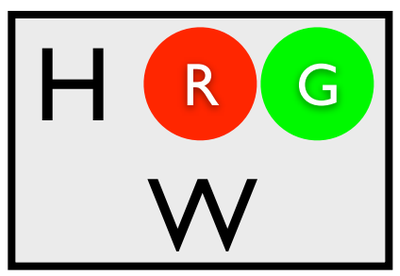
0.056



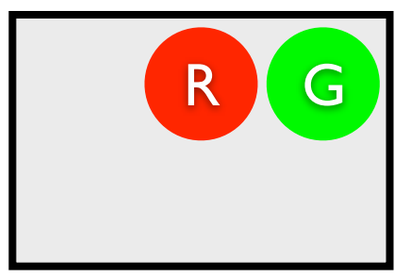
0.084



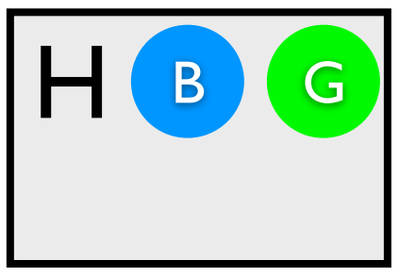
0.036



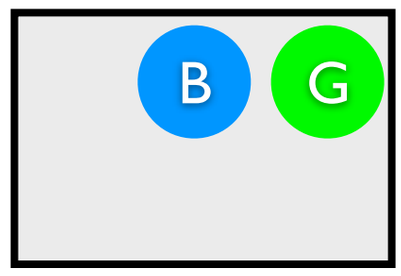
0.054



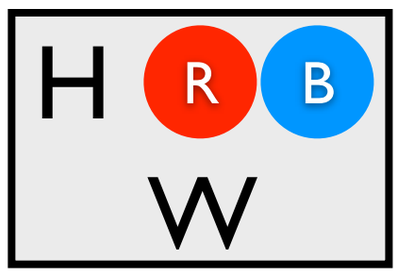
0.084



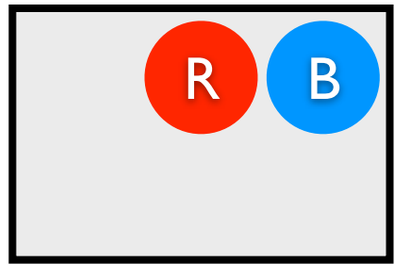
0.126



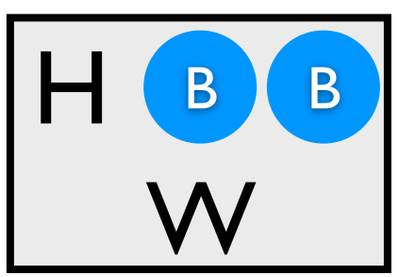
0.060



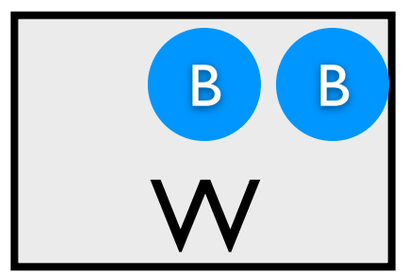
0.090



0.140



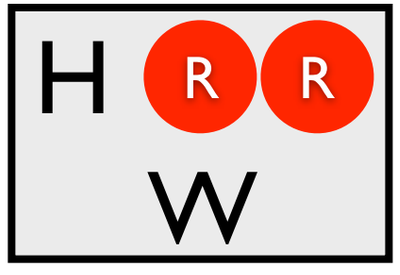
0.210



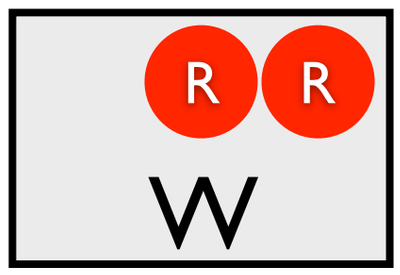
P(win)= $\sum$ =0.562

Marginal Probability

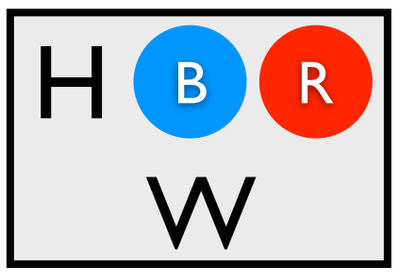
0.024



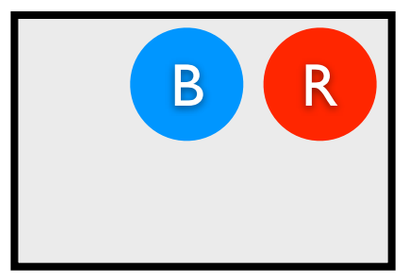
0.036



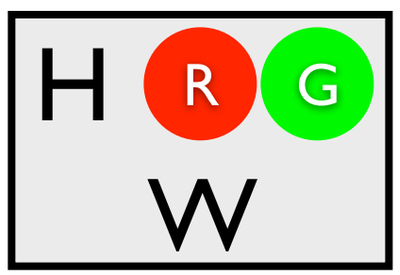
0.056



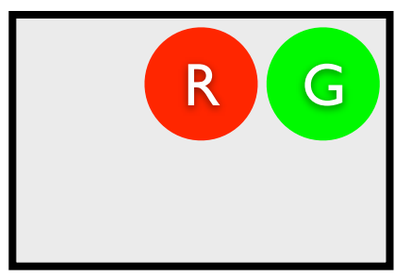
0.084



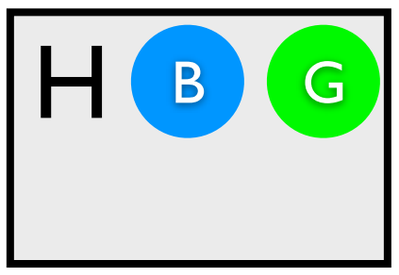
0.036



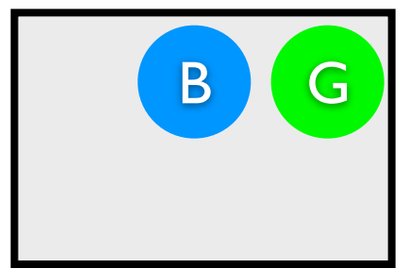
0.054



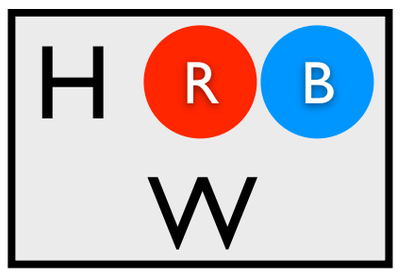
0.084



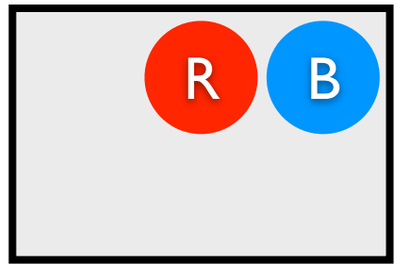
0.126



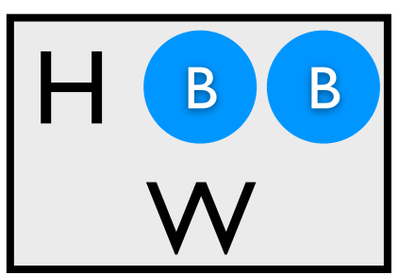
0.060



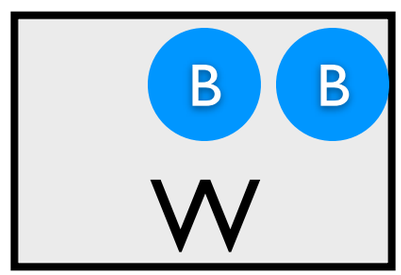
0.090



0.140



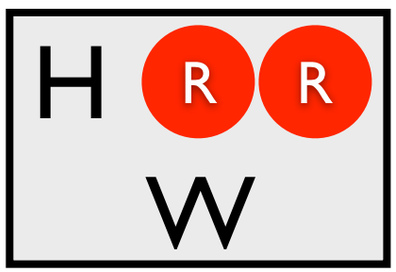
0.210



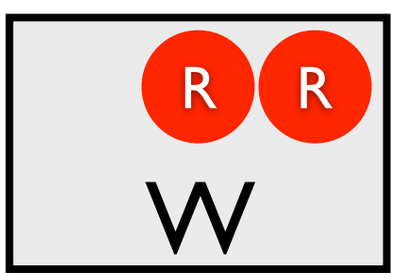
P(win|col(2,green))= ?

Conditional Probability

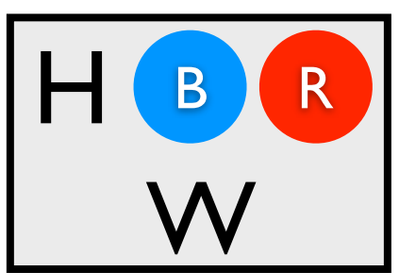
0.024



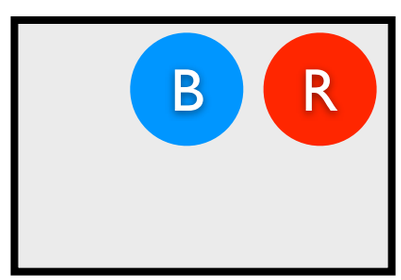
0.036



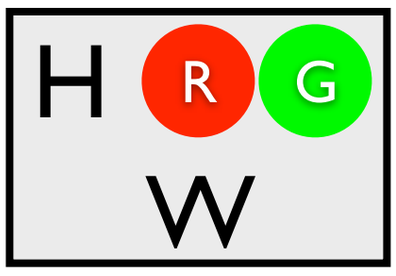
0.056



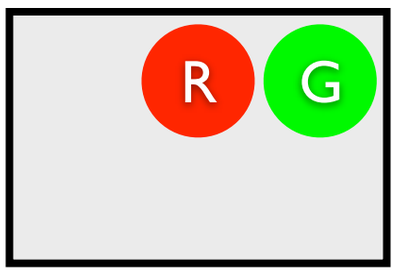
0.084



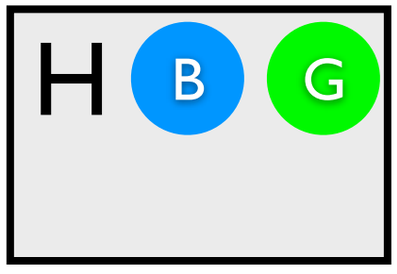
0.036



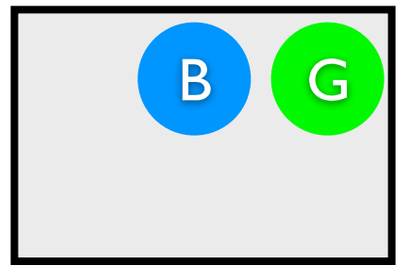
0.054



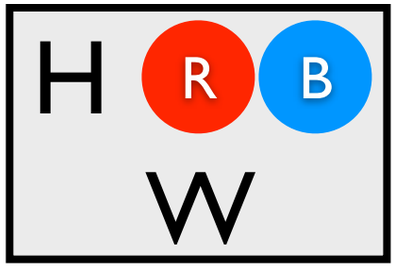
0.084



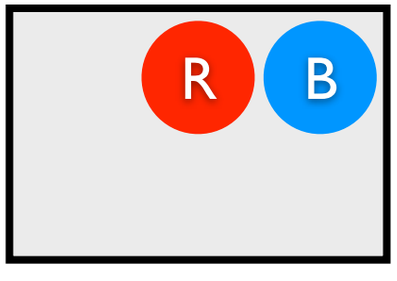
0.126



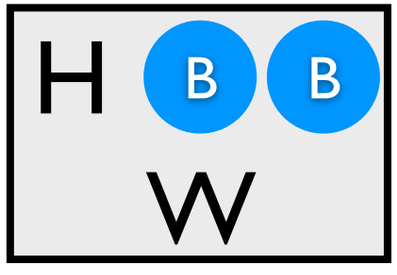
0.060



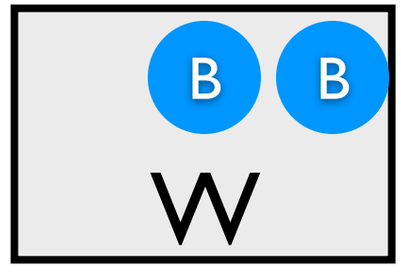
0.090



0.140



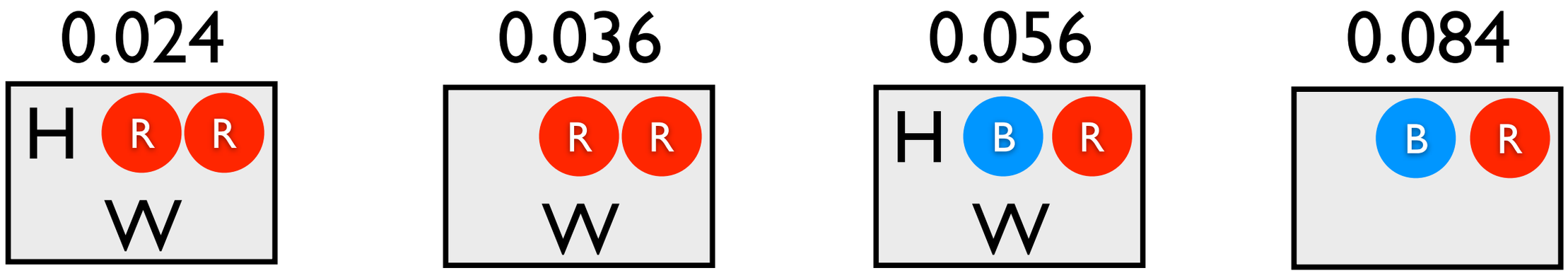
0.210



$$P(win|col(2,green)) = \Sigma/\Sigma$$

$$= P(win \wedge col(2,green))/P(col(2,green))$$

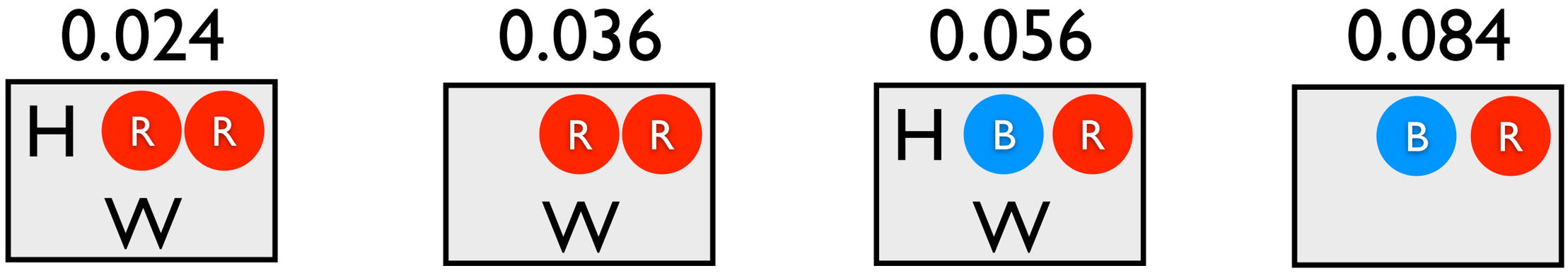
Conditional Probability



$$P(win|col(2,green)) = \Sigma/\Sigma$$

$$= P(win \wedge col(2,green))/P(col(2,green))$$

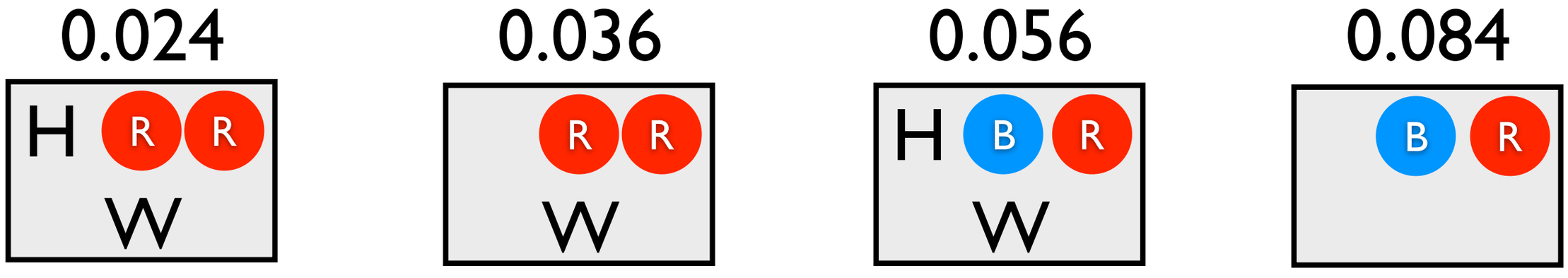
Conditional Probability



$$P(win|col(2,green)) = \Sigma/\Sigma$$

$$= 0.036/0.3 = 0.12$$

Conditional Probability



Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

sum over possible worlds where Q is true

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

sum over possible worlds where Q is true

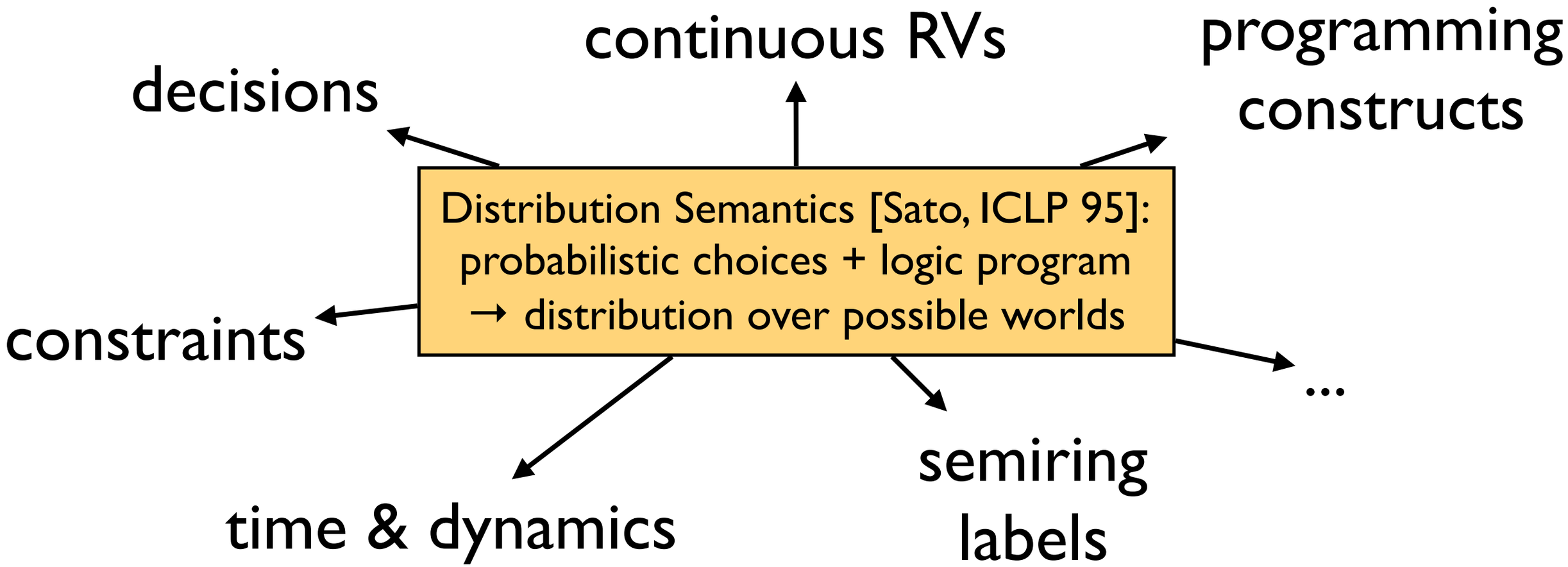
$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

probability of possible world

Extensions of basic PLP



Distributional Clauses (DC)

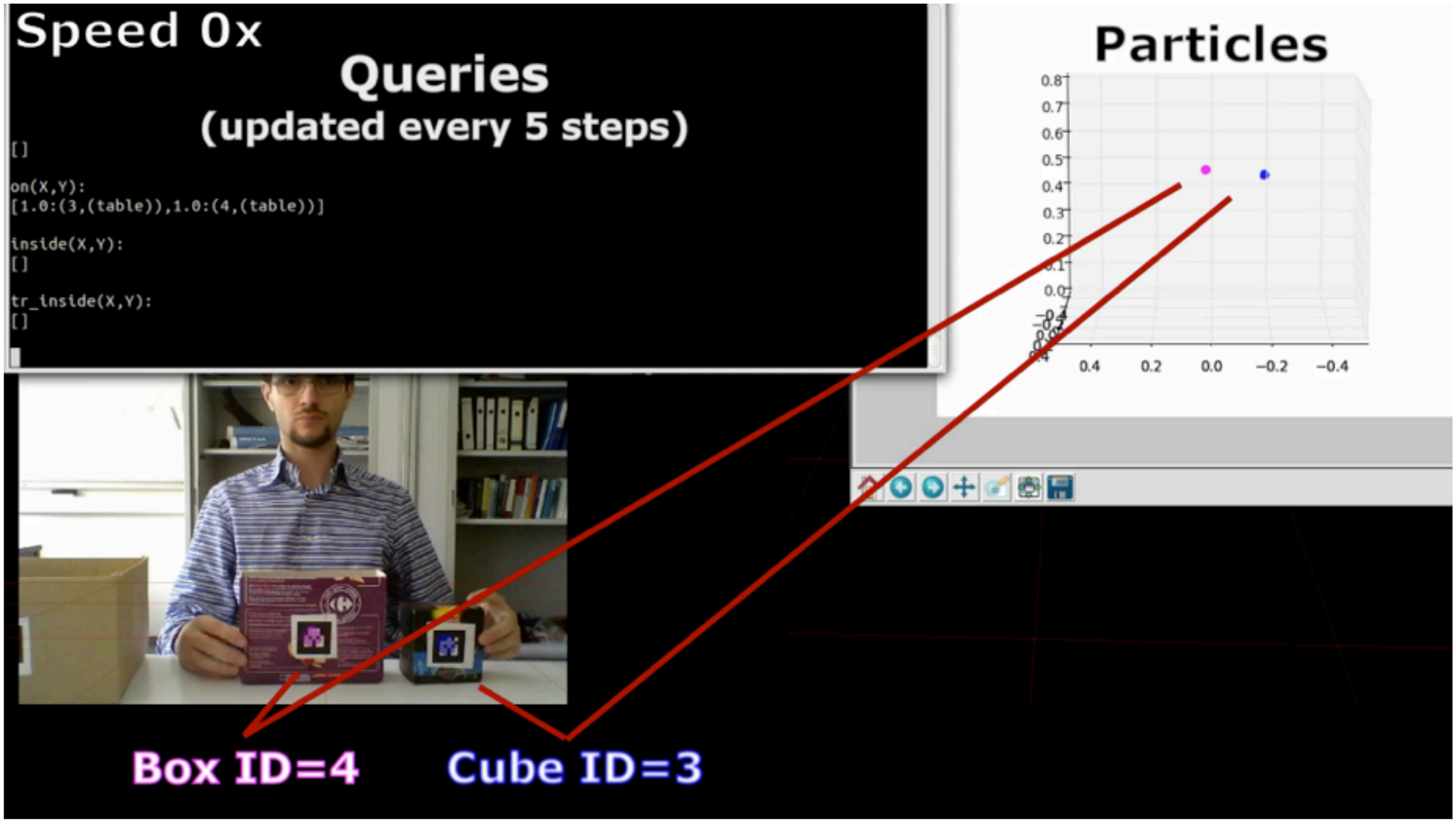
Discrete- and continuous-valued random variables

random variable with Gaussian distribution

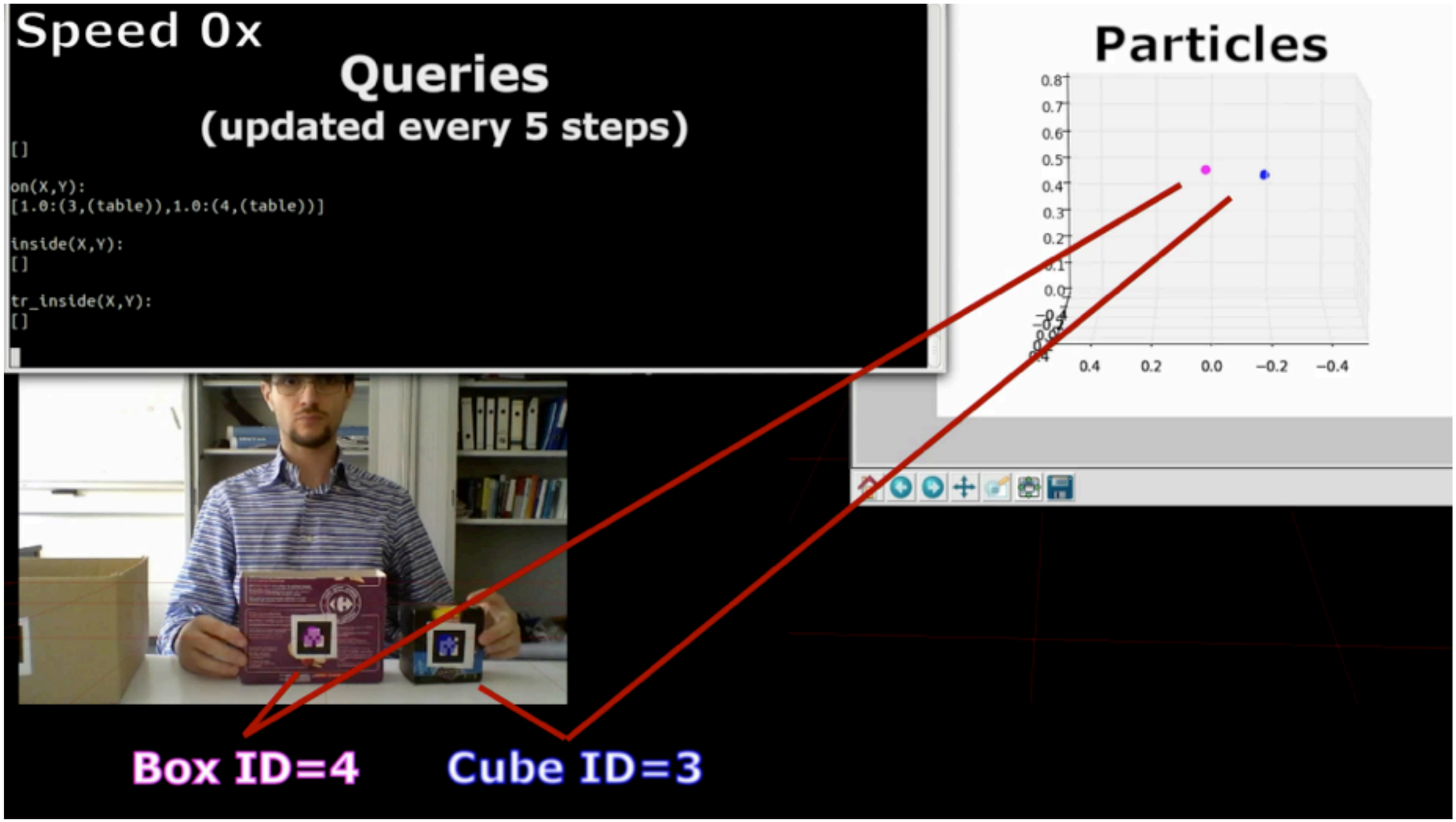
```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
        ≃length(OBot) ≥ ≃length(OTop),
        ≃width(OBot) ≥ ≃width(OTop).
```

comparing values of random variables

[Gutmann et al, TPLP 11;
Nitti et al, MLJ 16,17]


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13

https://dtai.cs.kuleuven.be/problog/tutorial.html

Click to go forward, hold to see history

ProbLog Home Download Publications Help People Interactive

probability.

- The express the dependence of the random variable *alarm* on its parents *burglary* and *earthquake*, we use one Prolog rule for every possible state of the parents.
    - The first rule covers the case in which burglary and earthquake are both true. The required rule is *alarm :- burglary, earthquake, p_alarm1* with *p_alarm1* an auxiliary atom defined by means of the probabilistic fact *0.9::p_alarm1*. The point of adding this atom is to ensure that the probability of *alarm* in this case will be 0.9 as required.
    - The second rule covers the case that burglary is true but earthquake is false. Note that earthquake being false is encoded using the "\+" symbol for negation (as in regular Prolog).
    - The third rule covers the case that burglary is false and earthquake is true.
    - The fourth case (burglary and earthquake are both false) does not require a rule. This is because, according to our Bayesian network, the probability of alarm is 0 in this case.

We obtain the following ProbLog program.

```
1  0.7::burglary.
2  0.2::earthquake.
3  0.9::p_alarm1.
4  0.8::p_alarm2.
5  0.1::p_alarm3.
6
7  alarm :- burglary, earthquake, p_alarm1.
8  alarm :- burglary, \+earthquake, p_alarm2.
9  alarm :- \+burglary, earthquake, p_alarm3.
10
11 evidence(alarm,true).
12
13 query(burglary).
14 query(earthquake).
15
```

Evaluate

Results ...

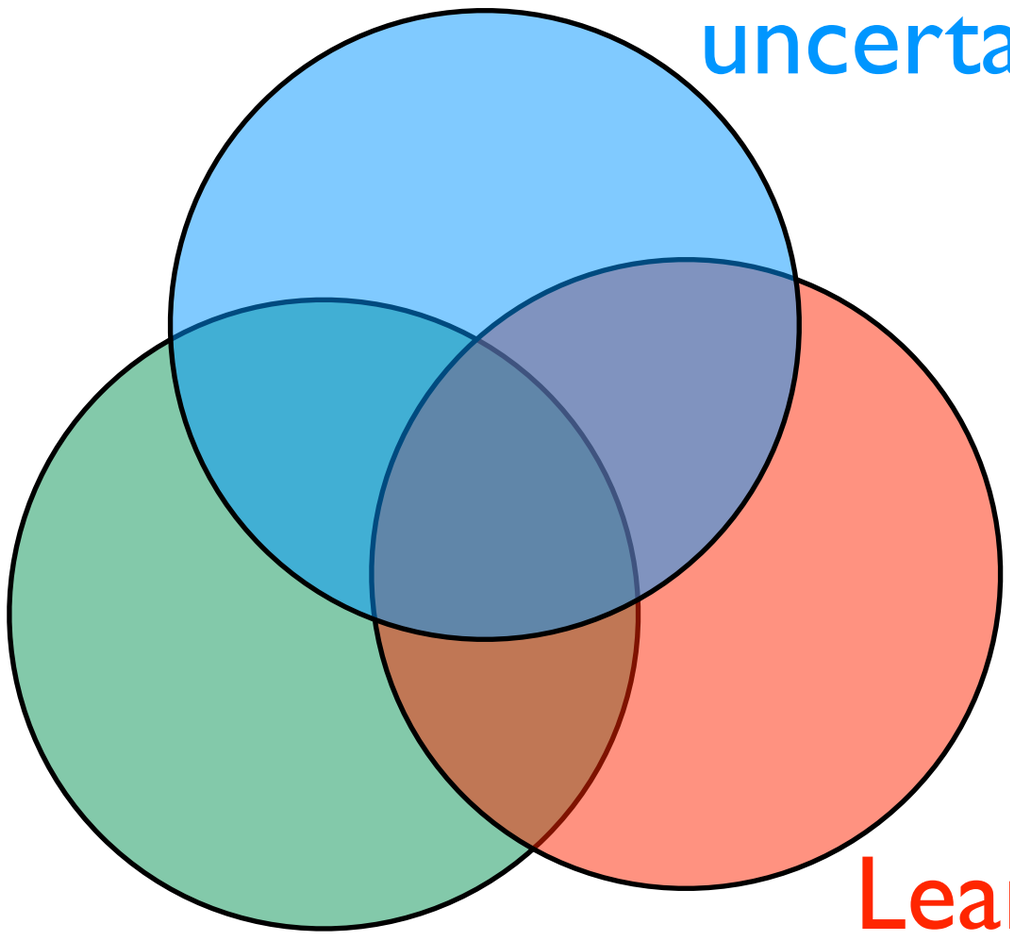
[Suciu et al 2011]

Probabilistic Databases

Dealing with uncertainty

Reasoning with relational data

Learning



[Suciu et al 2011]

Probabilistic Databases

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

bornIn

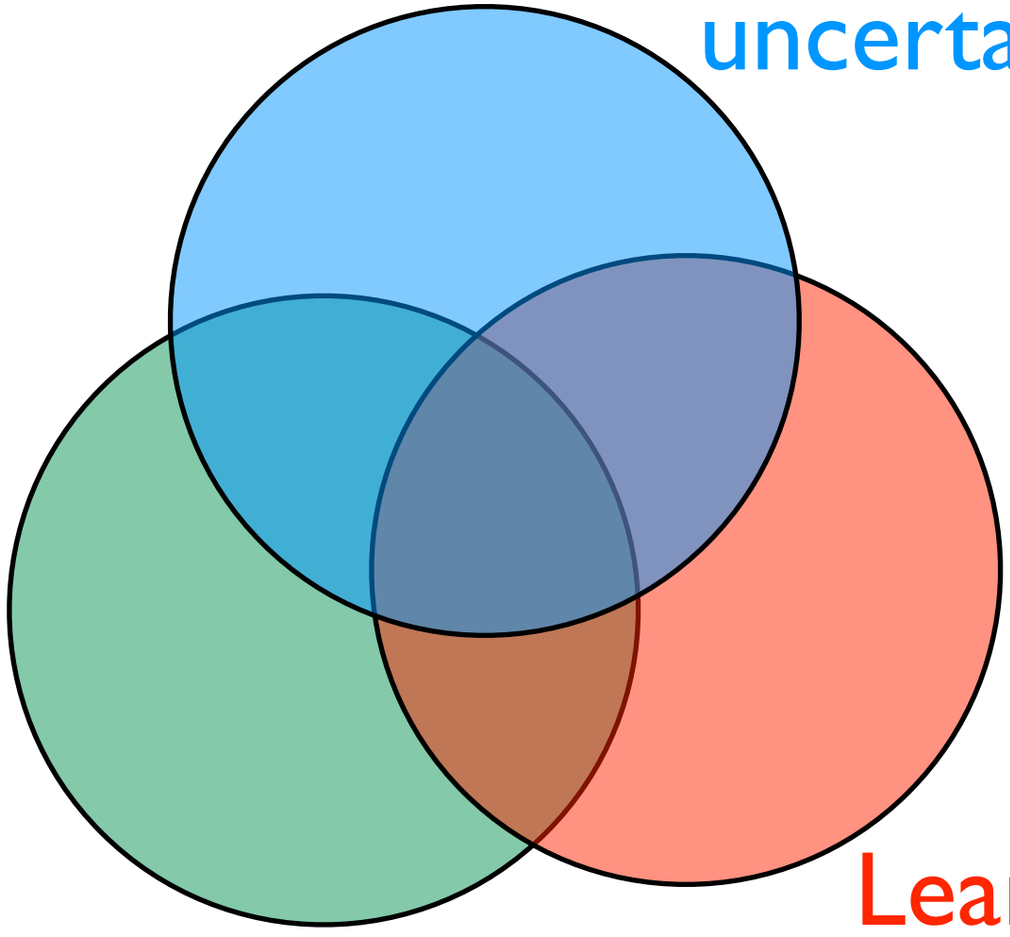
person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database

Dealing with uncertainty



Learning

[Suciu et al 2011]

Probabilistic Databases

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

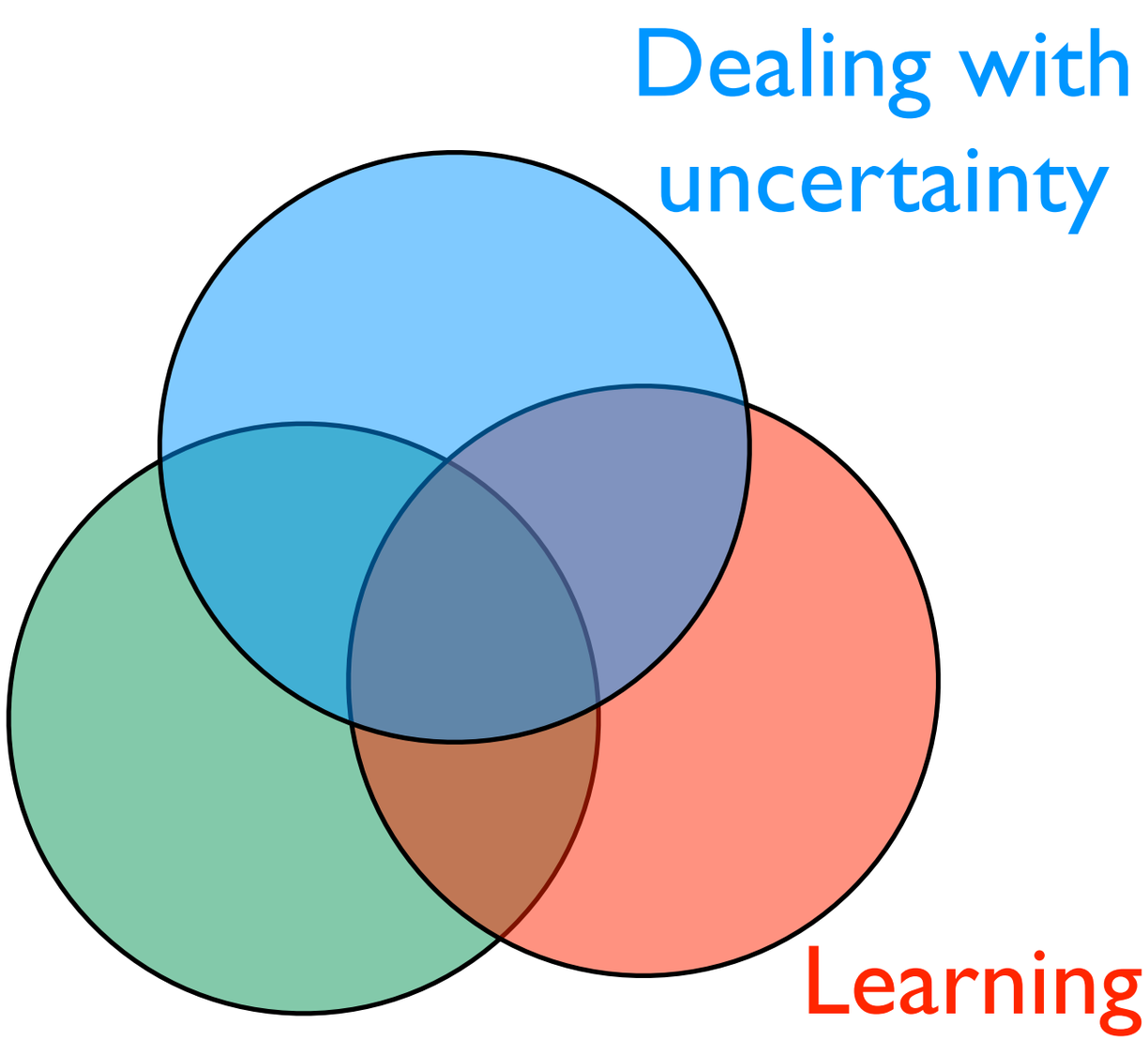
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



[Suciu et al 2011]

Probabilistic Databases

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random variables

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

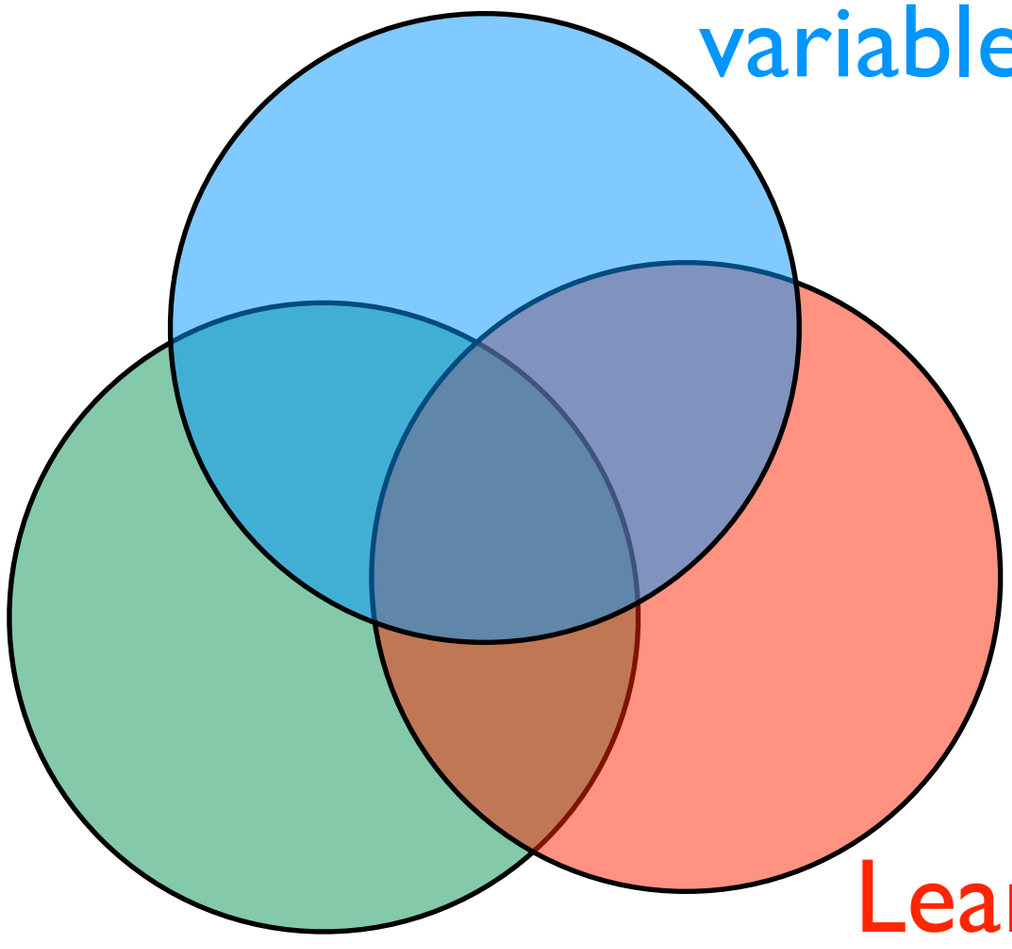
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Learning

[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

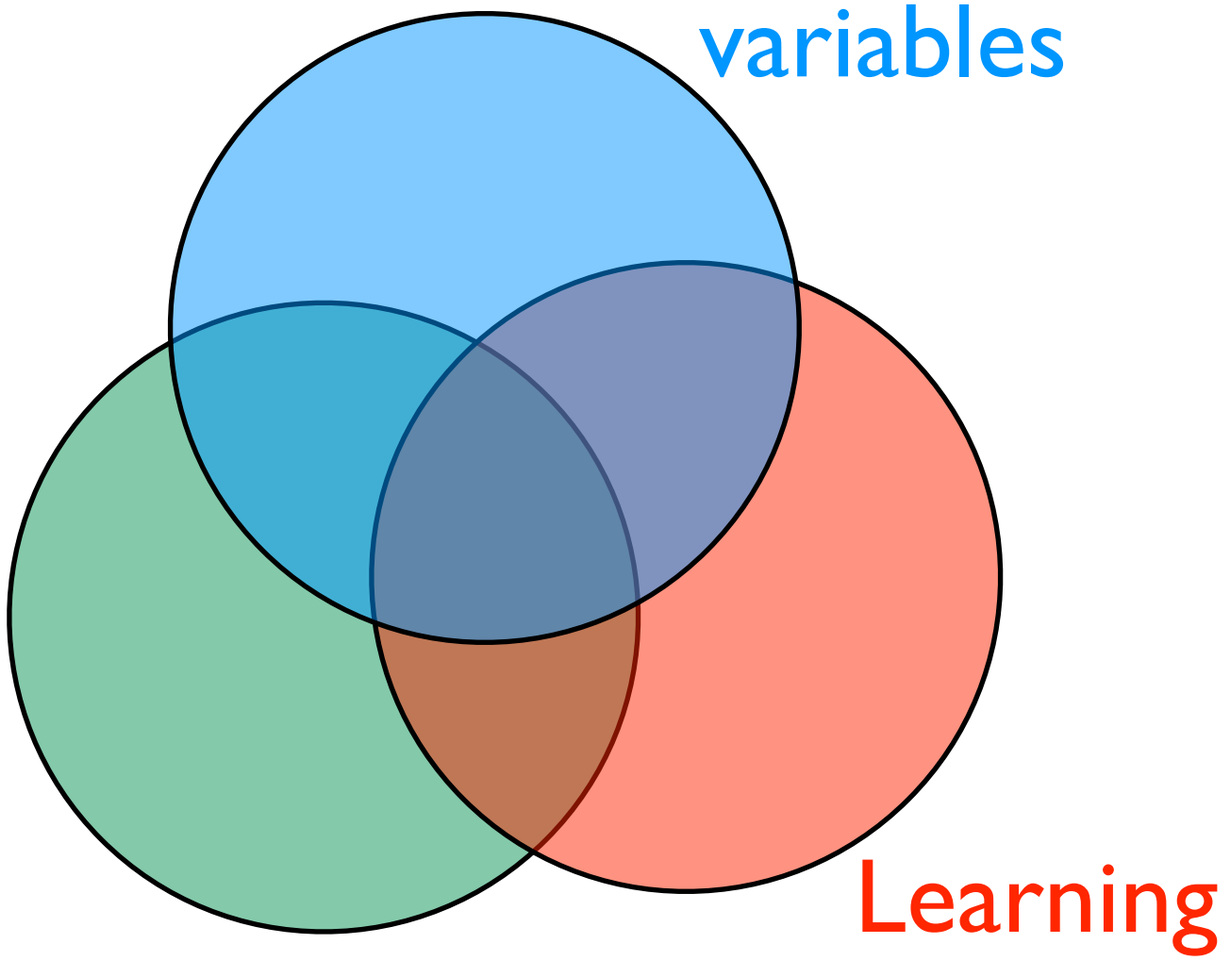
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
		0,9
		0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random variables

probabilistic tables + database queries
→ distribution over possible worlds

```
select
from bornIn x, cityIn y
where x.city=y.city
```

one world

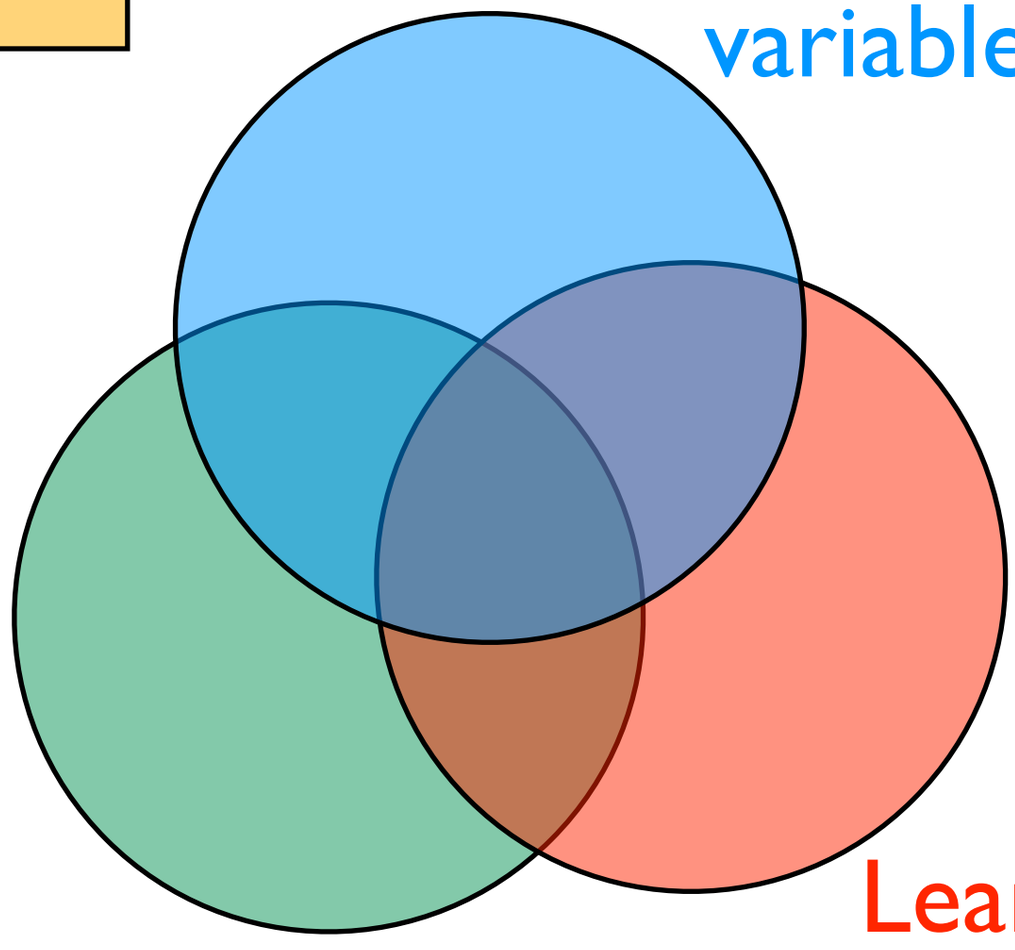
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database

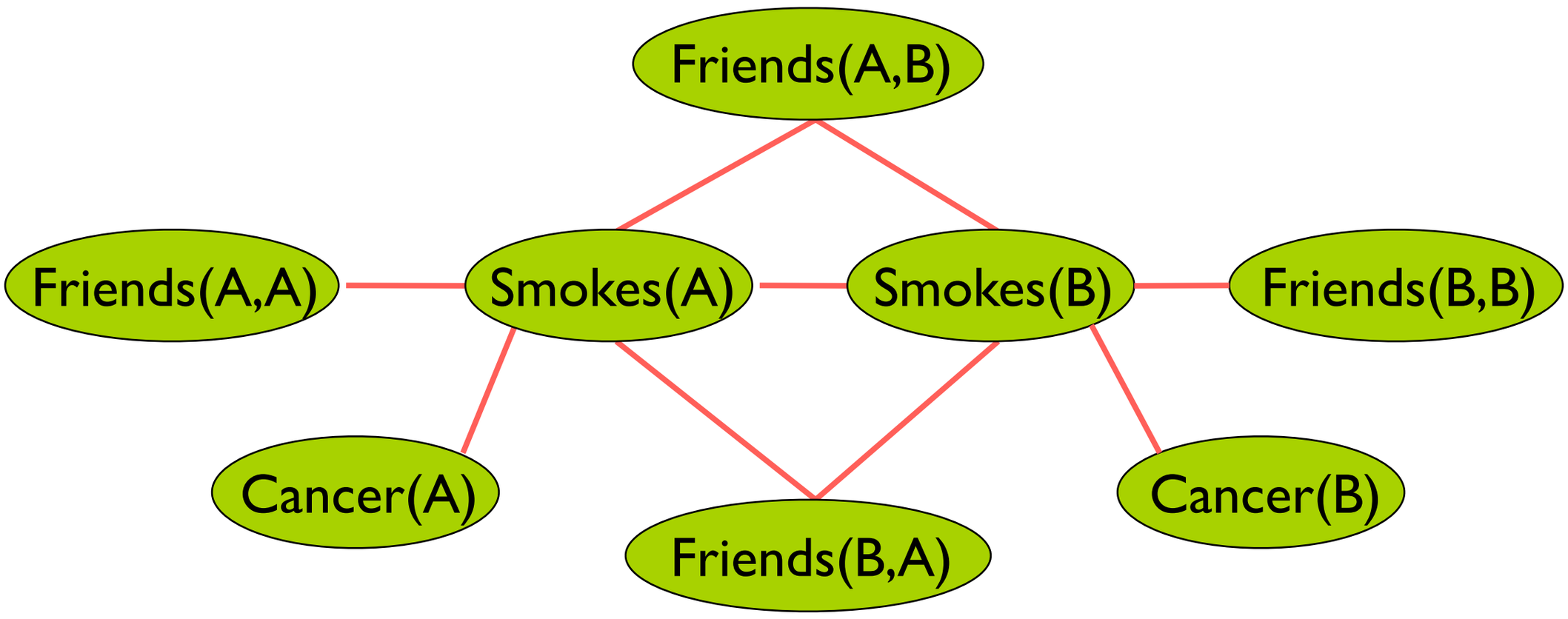


Learning

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Suppose we have two constants: **Anna** (A) and **Bob** (B)



slide by Pedro Domingos

Markov Logic Networks

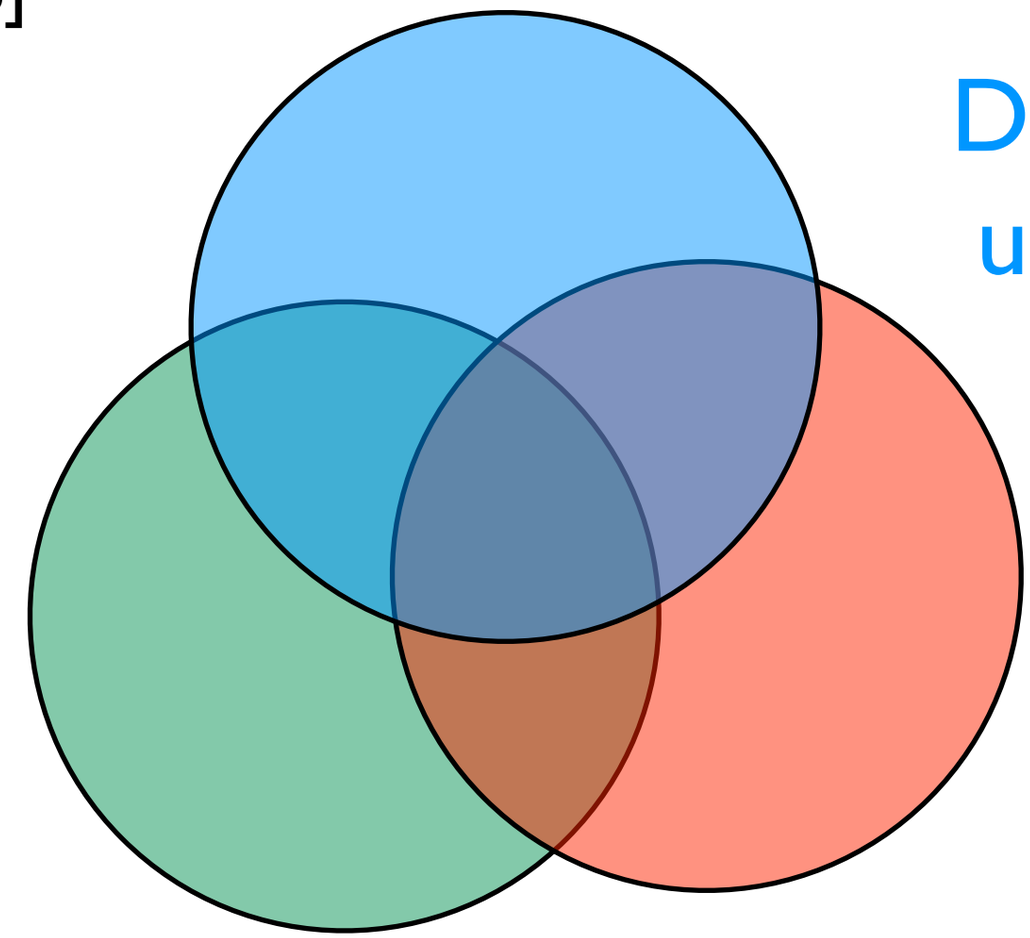
Some key differences :

- MLNs are not a programming language
- MLNs are generating an undirected graphical model;
    - PPs are directed
- MLNs allows for multiple models — p v q (needs max entropy to assign prob. to models {p}, {q}, {p,q});
    - Probabilistic Logic Programs use least-fix point semantics
- MLNs / first order logic cannot express transitive closure of relation; logic programs can
    - p(X,Y) :- p(X,Z), p(Z,Y)

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

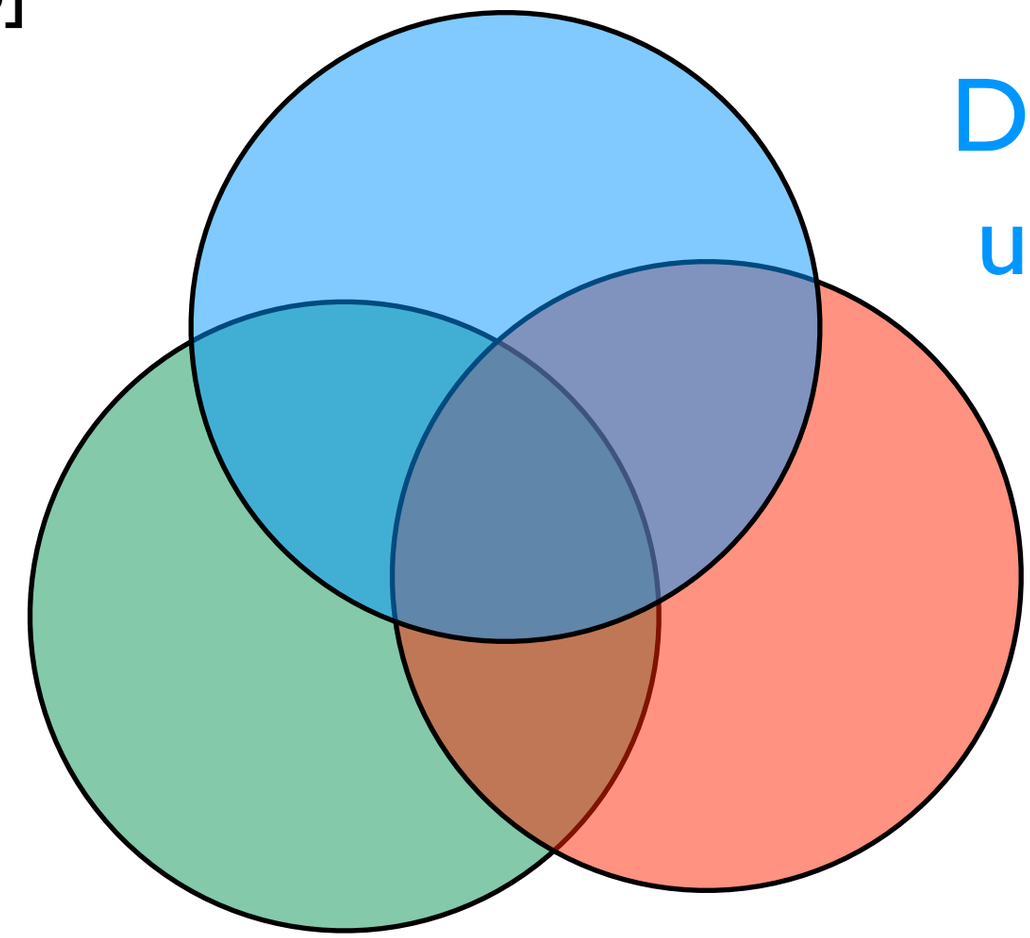
Reasoning with relational data

Learning

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

functional programming

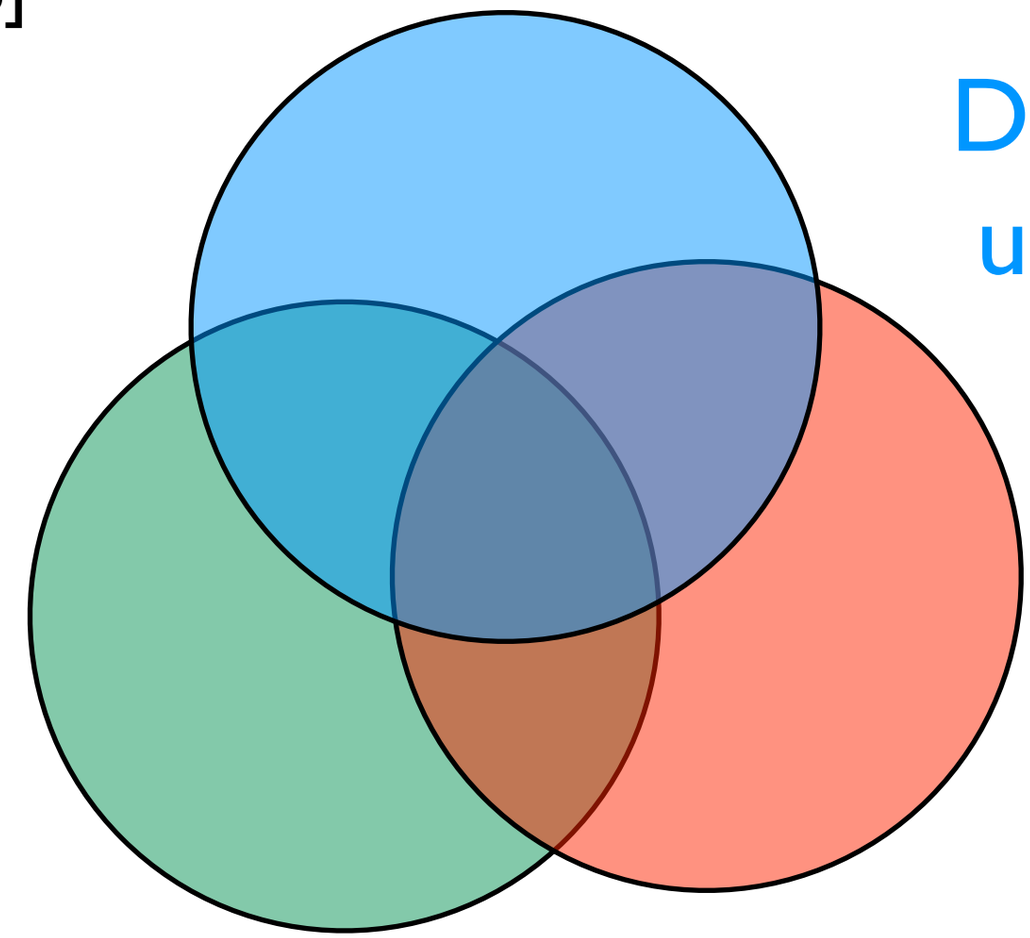
Learning

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church

probabilistic functional programming

[Goodman et al, UAI 08]



one execution

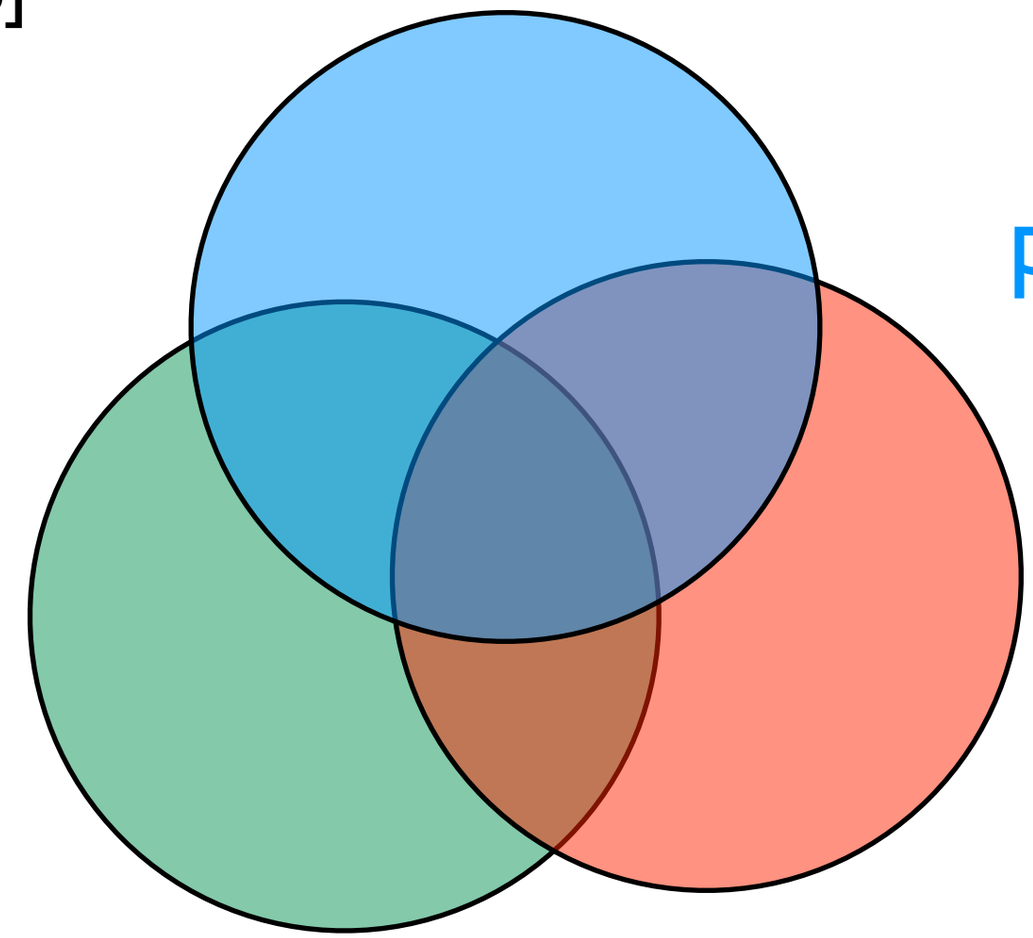
```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church

probabilistic functional programming

[Goodman et al, UAI 08]

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



random primitives

functional programming

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

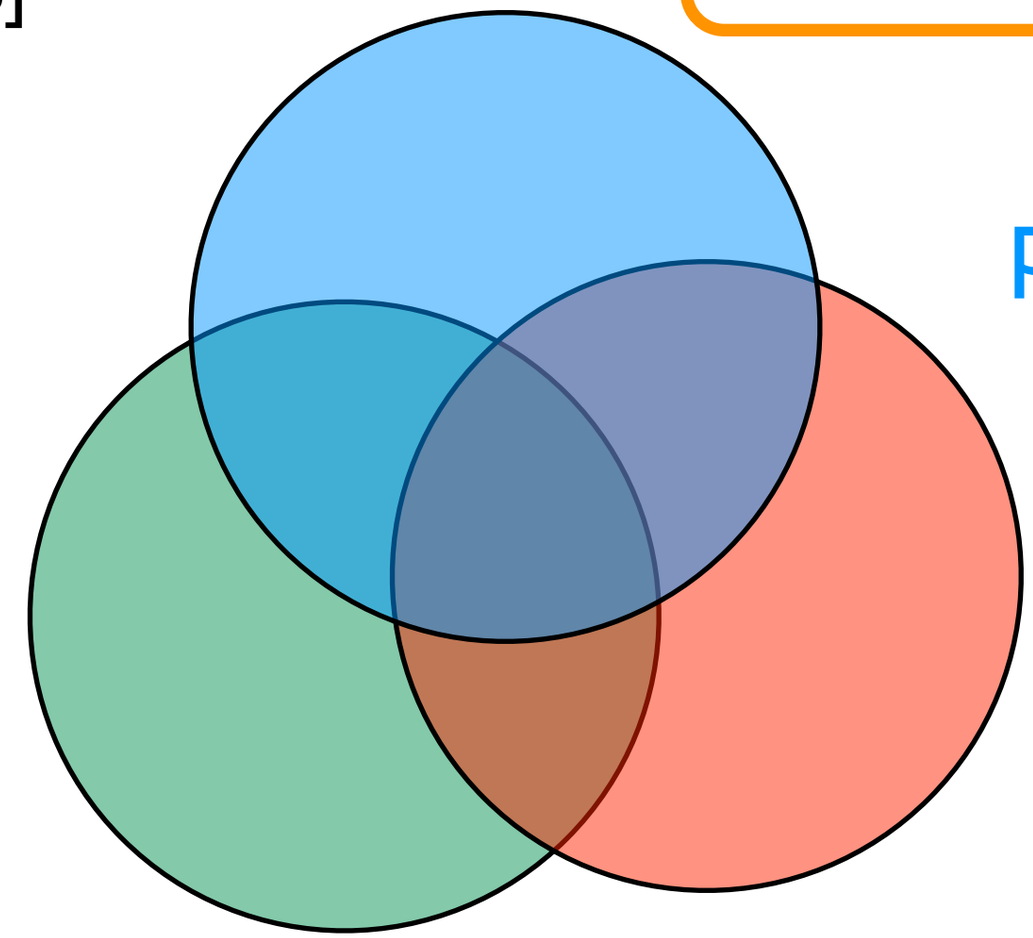
Church

probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



random primitives

functional programming

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

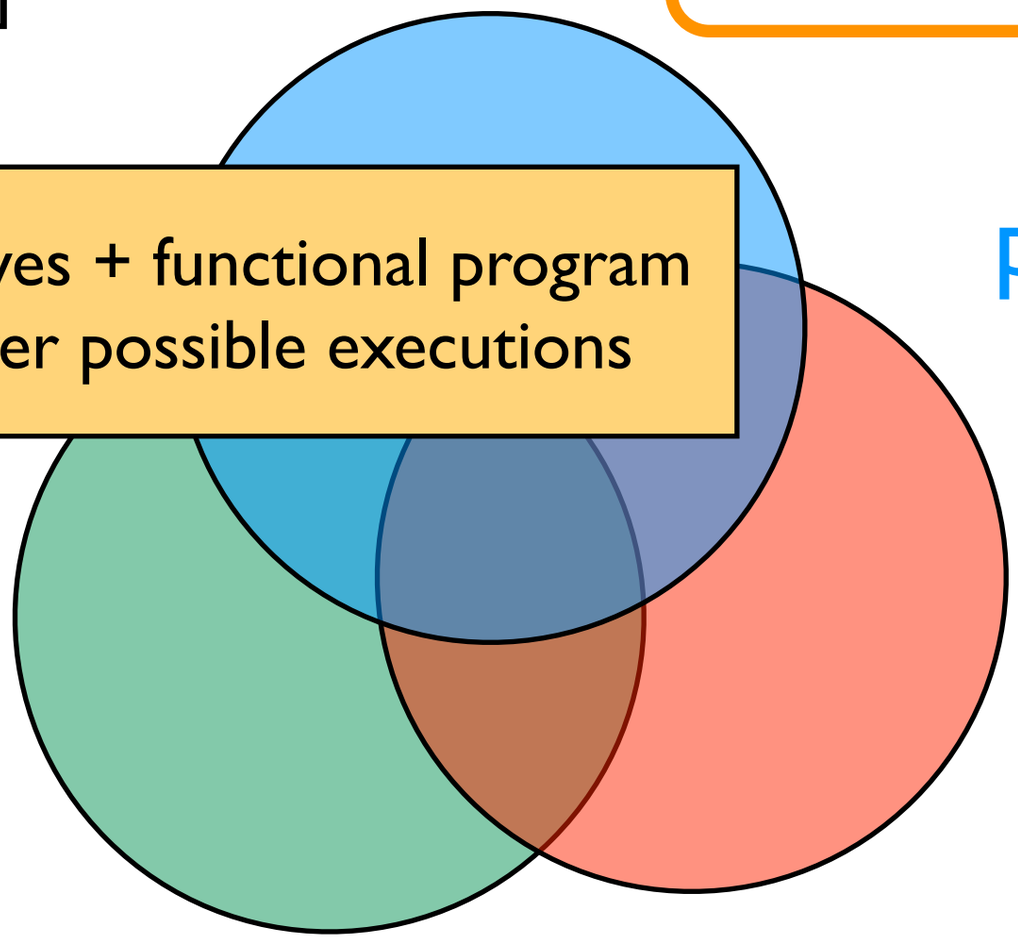
Church

probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



probabilistic primitives + functional program
→ distribution over possible executions

random primitives

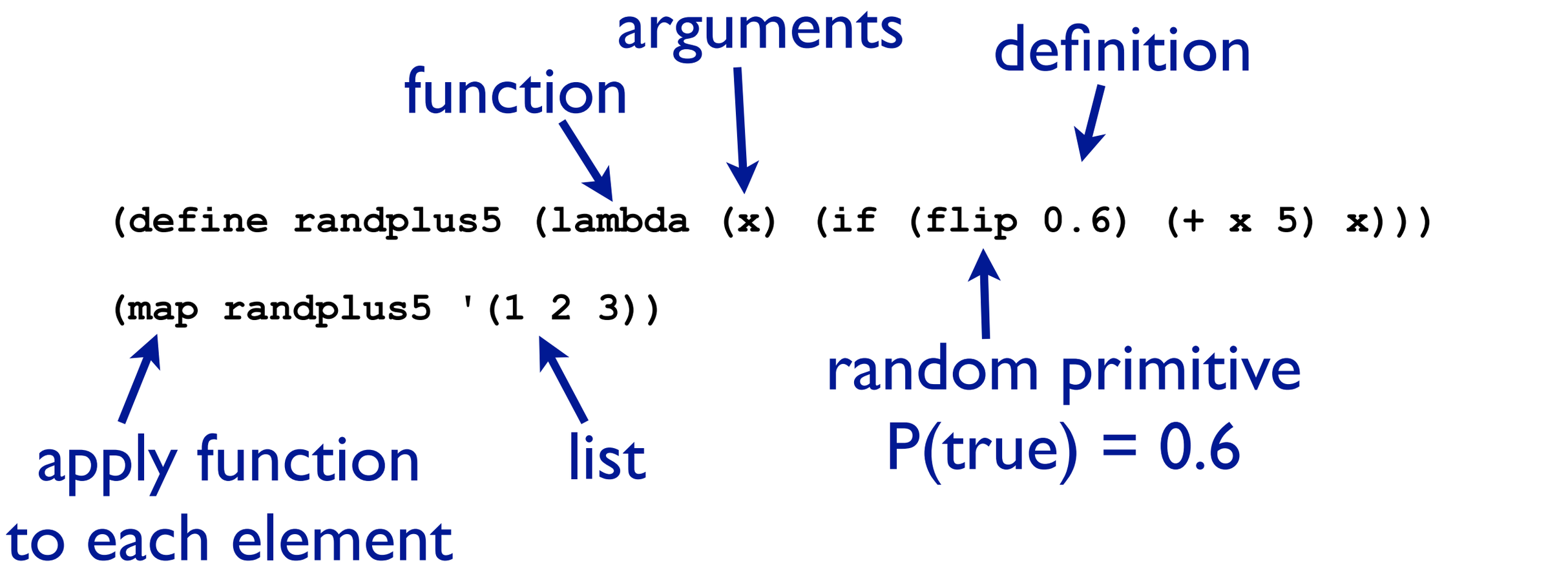
functional programming

Learning

one execution

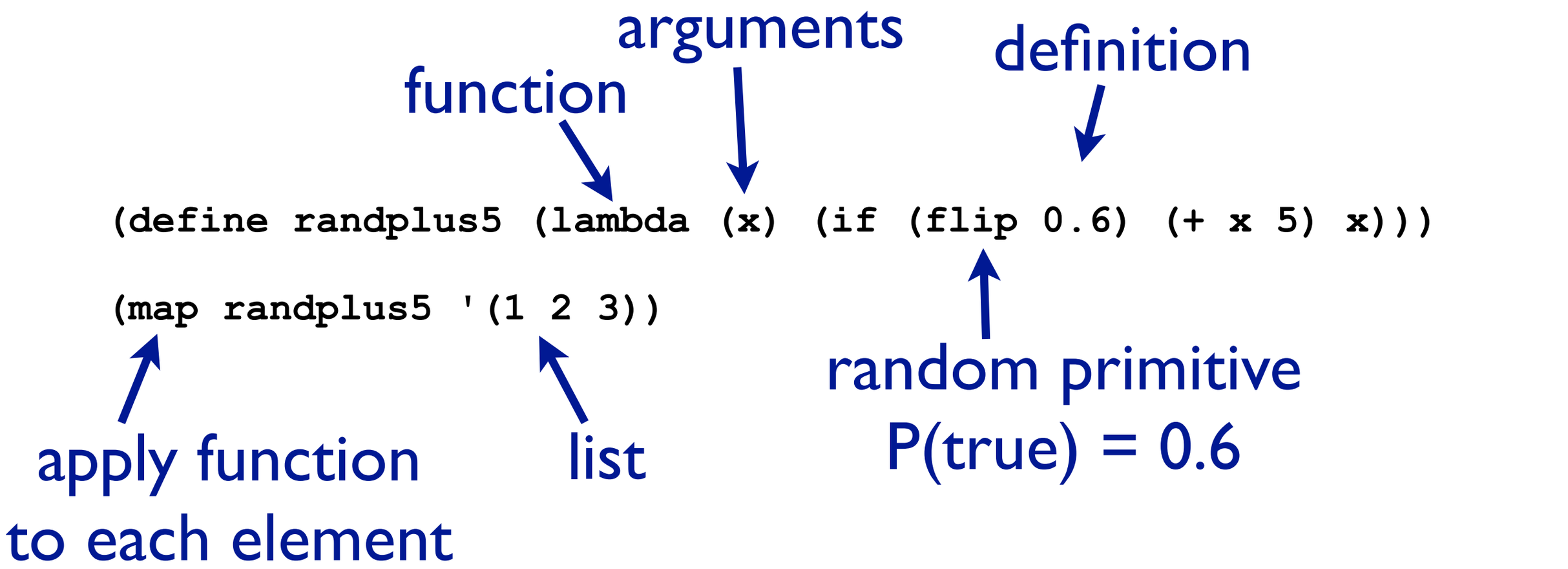
```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church Example



(1 2)	(1 7)	(6 2)	(6 7)
0.16	0.24	0.24	0.36

Church Example

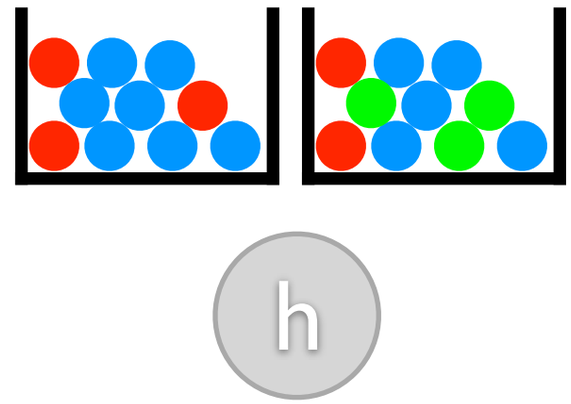


sampling also supports continuous RVs, e.g.,
`(* (gaussian 0 1) (gaussian 0 1))`

(1 2)	(1 7)	(6 2)	(6 7)
0.16	0.24	0.24	0.36

Church by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

Roadmap

1. Modeling
2. Inference
3. Learning
4. Applications

... with some detours on the way

PART II : Inference

Inference / Reasoning

- Most of the work in PP and StarAI is on inference
  - It is (#P)-hard (complexity wise)
    - Many inference methods
      - exact, approximate, sampling and lifted …
- Inference is the key to learning

Dichotomy of UCQ Evaluation

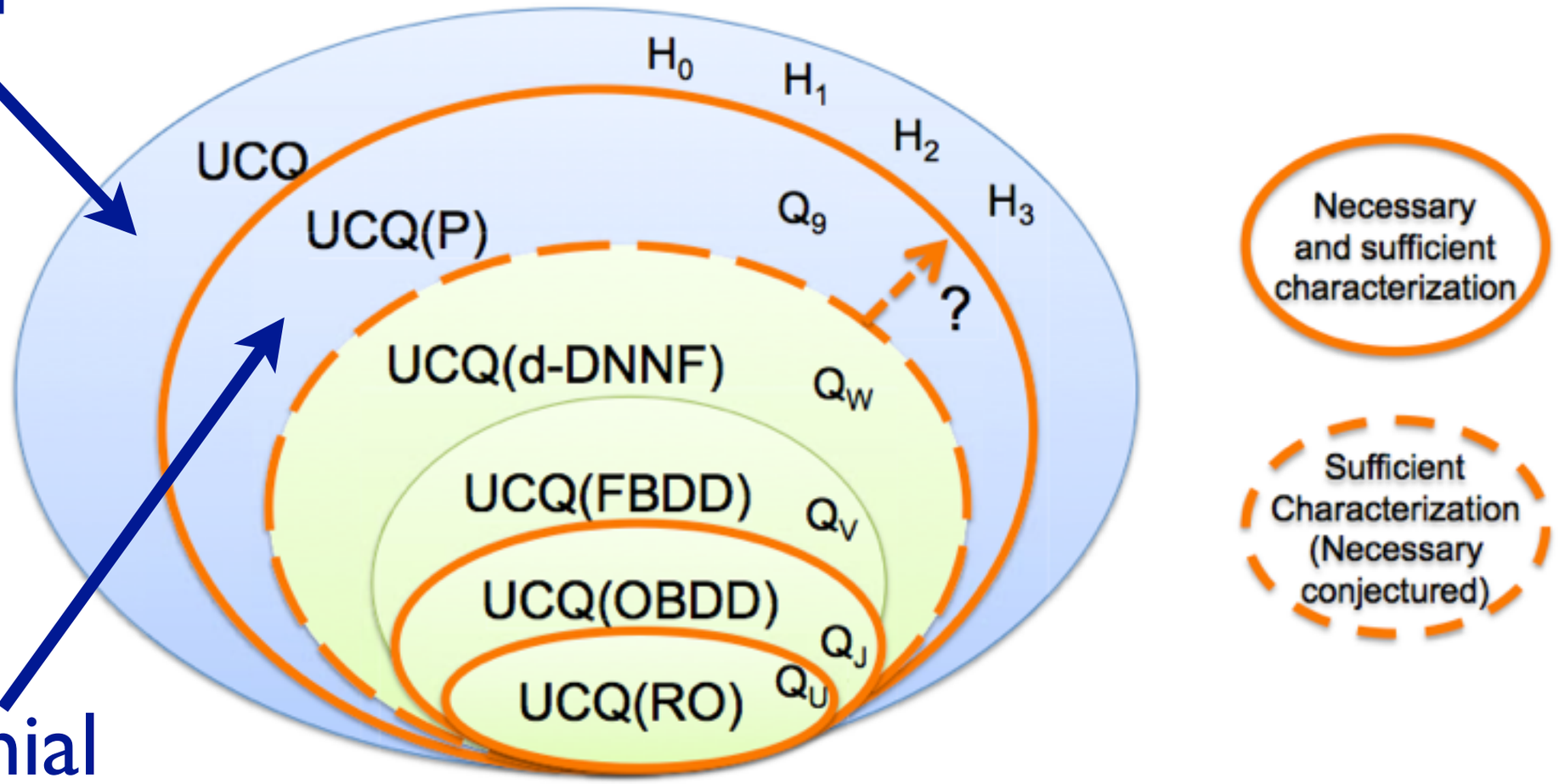
- Union of Conjunctive Queries
  ≈ Datalog without recursion and negation
- Theorem: UCQ evaluation is either polynomial in database size or #P-hard

Dichotomy of UCQ Evaluation

- Union of Conjunctive Queries
  ≈ Datalog without recursion and negation
- Theorem: UCQ evaluation is either polynomial in database size or #P-hard

counting version of NP decision problems, e.g., model counting

#P-hard

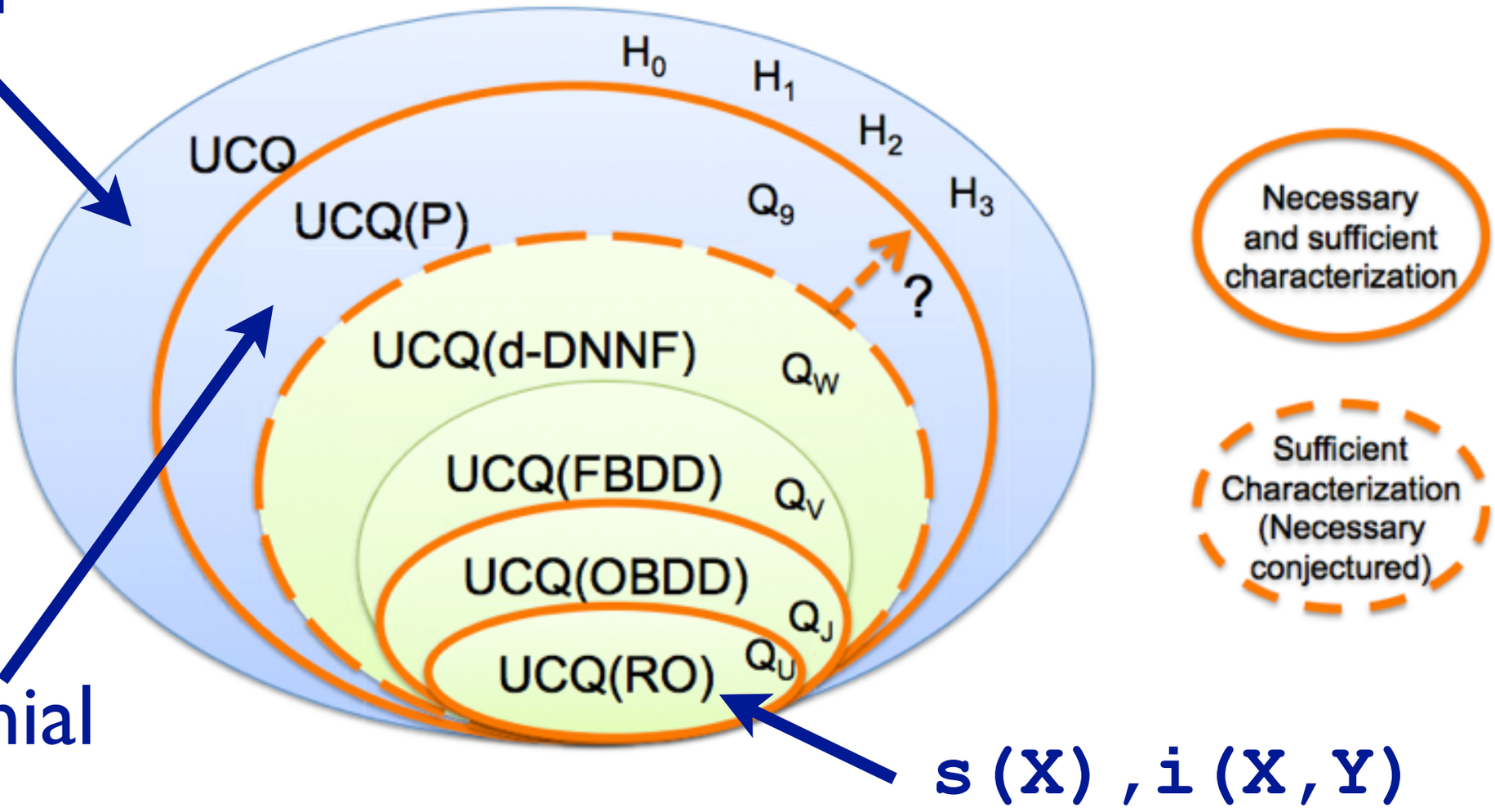


polynomial

Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]

#P-hard



polynomial

s(X),i(X,Y)

Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]

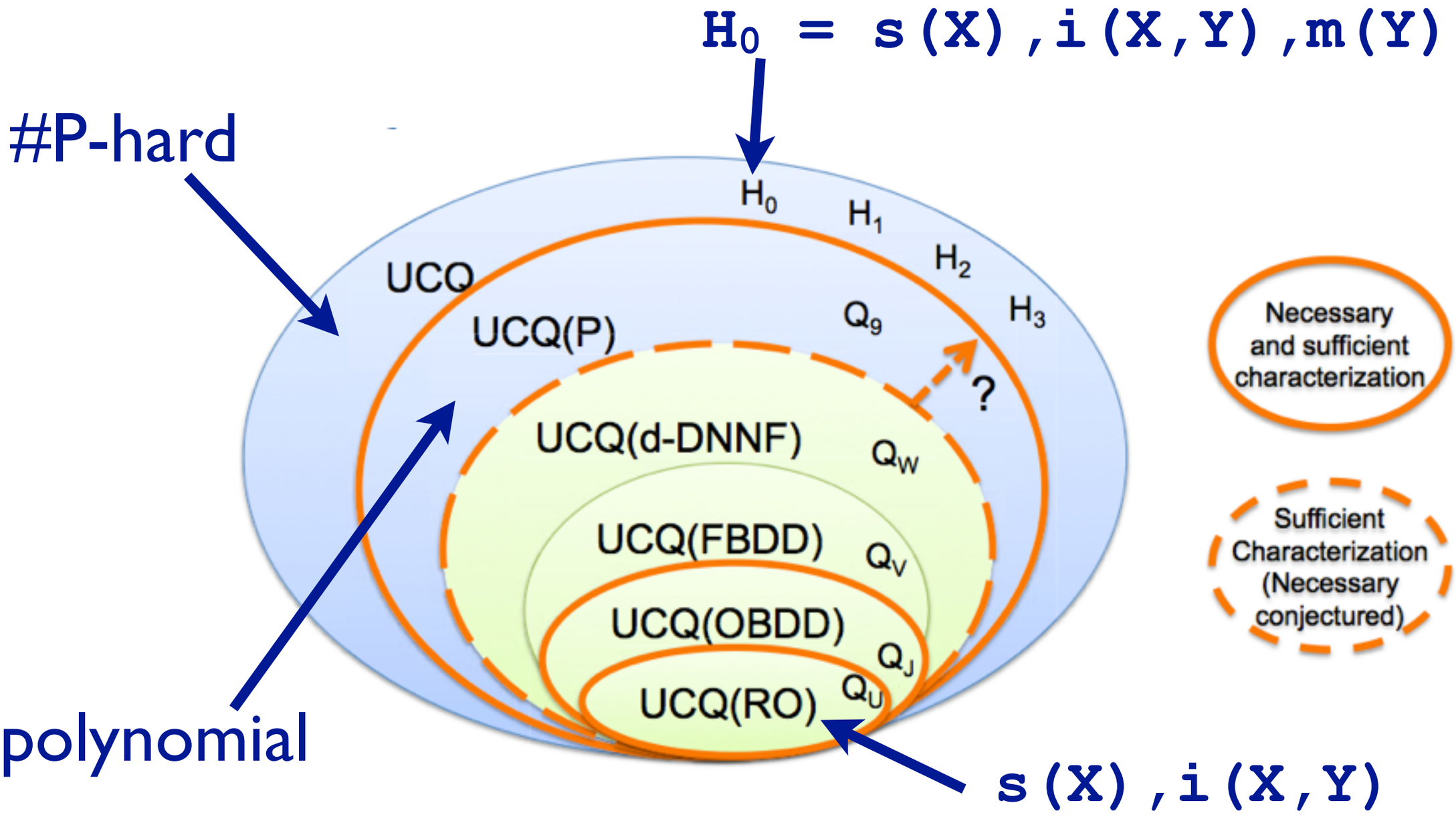


Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

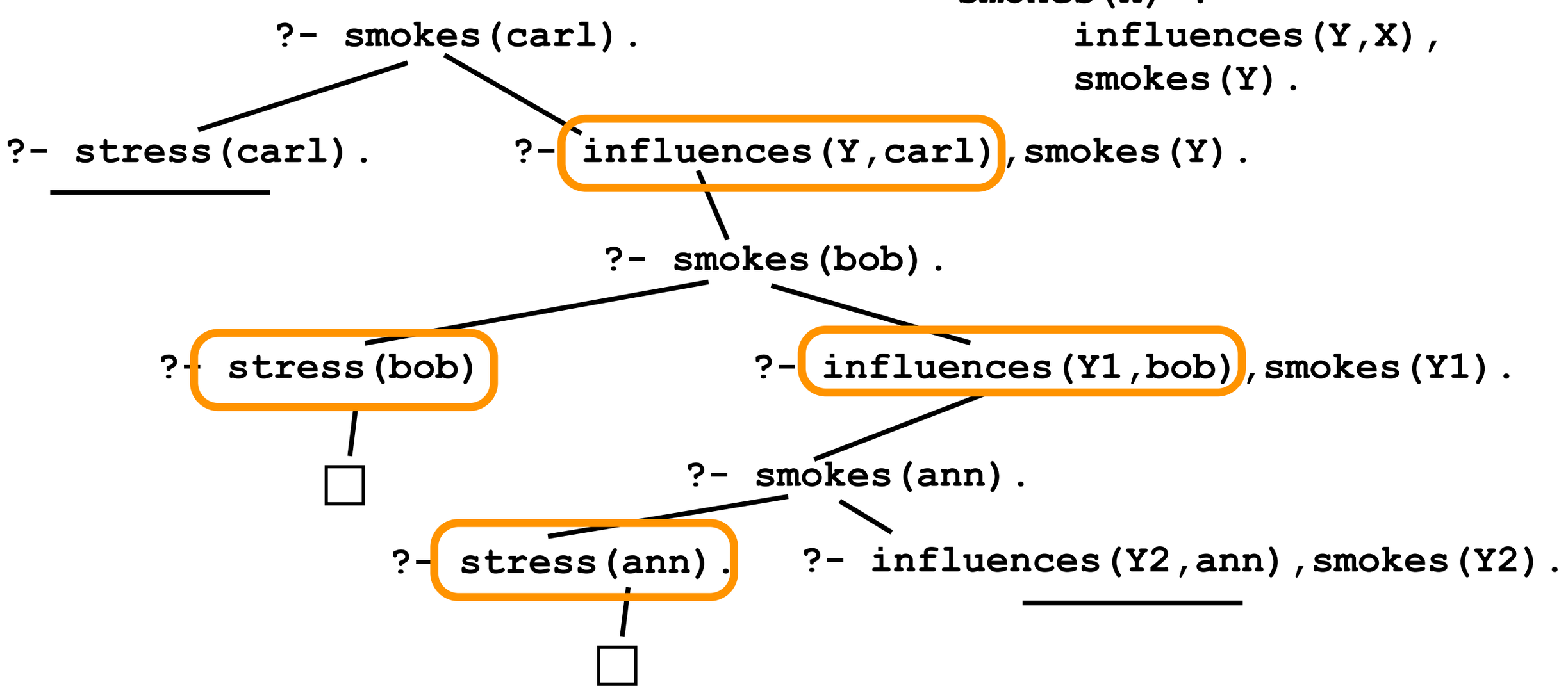
Fig. from [Suciu et al 2011]

Two Steps

- **Logical inference -**
  - proofs or model theoretic …
  - *Result: Weighted Model Counting problem*
- **Probabilistic propositional inference —**
  - Backtracking search — DPLL, VE, RC based
  - Knowledge Compilation
- **Advanced** — lifted inference

Proofs in Pro(b)Log

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



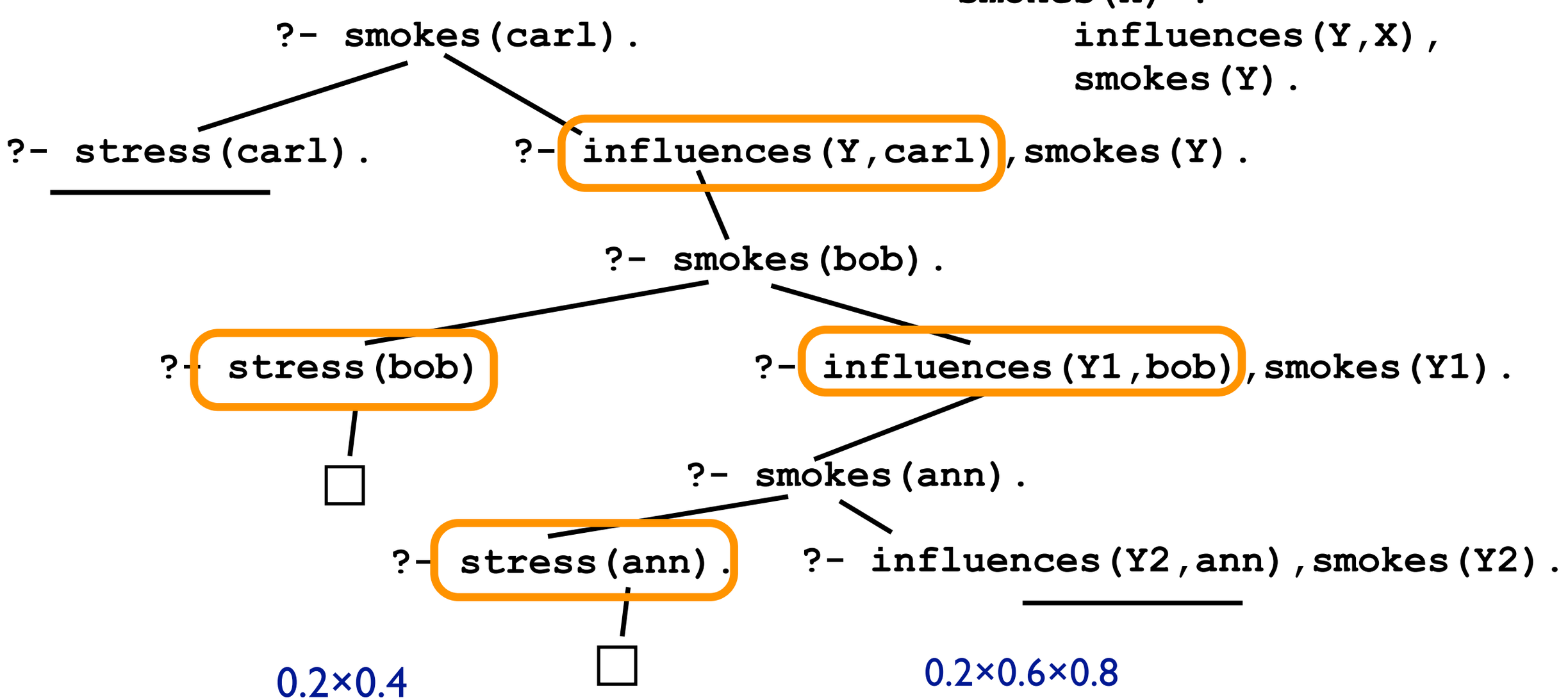

```
influences(bob,carl)
    & stress(bob)
```

OR

```
influences(bob,carl)
  & influences(ann,bob)
      & stress(ann)
```

Proofs in Pro(b)Log

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



0.2×0.4
= 0.08

`influences(bob,carl)`
`& stress(bob)`

OR

0.2×0.6×0.8
= 0.096

`influences(bob,carl)`
`& influences(ann,bob)`
`& stress(ann)`

Answer Set Programming

```
?- smokes(carl).
```

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

```
smokes(carl):- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

```
smokes(carl):- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

```
smokes(carl):- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

- Convert to propositional logic formula

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

```
smokes(carl):- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

- Convert to propositional logic formula

may require loop-breaking

$$sm(c) \leftrightarrow (i(b,c) \wedge sm(b))$$
$$sm(b) \leftrightarrow (st(b) \vee (i(a,b) \wedge sm(a)))$$
$$sm(a) \leftrightarrow st(a)$$

Answer Set Programming

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning and grounding

```
smokes(carl):- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

- Convert to propositional logic formula

may require loop-breaking

$$sm(c) \leftrightarrow (i(b,c) \wedge sm(b))$$
$$sm(b) \leftrightarrow (st(b) \vee (i(a,b) \wedge sm(a)))$$
$$sm(a) \leftrightarrow st(a)$$

Disjoint-Sum-Problem

possible worlds

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
```

... **influences(bob,carl) & stress(bob)**

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
```

... **influences(bob,carl) & stress(bob)**

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)	0.0576
infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)	0.0384
infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)	0.0256
infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)	0.0096
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)	0.0064
... **influences(bob,carl) & stress(bob)**	∑ = 0.1376

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

solution: knowledge compilation

`infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)`	0.0576
`infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)`	0.0384
`infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)`	0.0256
`infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)`	0.0096
`infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)`	0.0064
`...` `influences(bob,carl) & stress(bob)`	$\sum = 0.1376$

sum of proof probabilities: 0.096+0.08 = 0.1760

SAT

Satisfiability of CNF formula

$$(A \vee B) \wedge (\neg B \vee C)$$

$$\bigvee_{A,B,C \in \{true,false\}} (A \vee B) \wedge (\neg B \vee C)$$

$$\bigvee_{A,B,C \in \{true,false\}} c_1(A,B) \wedge c_2(B,C)$$

$$\bigvee_{A,B,C \in \{true,false\}} \bigwedge_i c_i(A,B,C)$$

influences(bob,carl) & stress(bob) **OR** **influences(bob,carl) & influences(ann,bob) & stress(ann)**

#SAT

Counting the number of satisfying assignments

$$\sum_{A,B,C \in \{true,false\}} true?(A \vee B) \times true?(B \vee C)$$

$$\sum_{A,B,C \in \{true,false\}} \prod_i c_i(A,B,C)$$

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for instance …

w(f) = p

w(not f) = 1−p

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

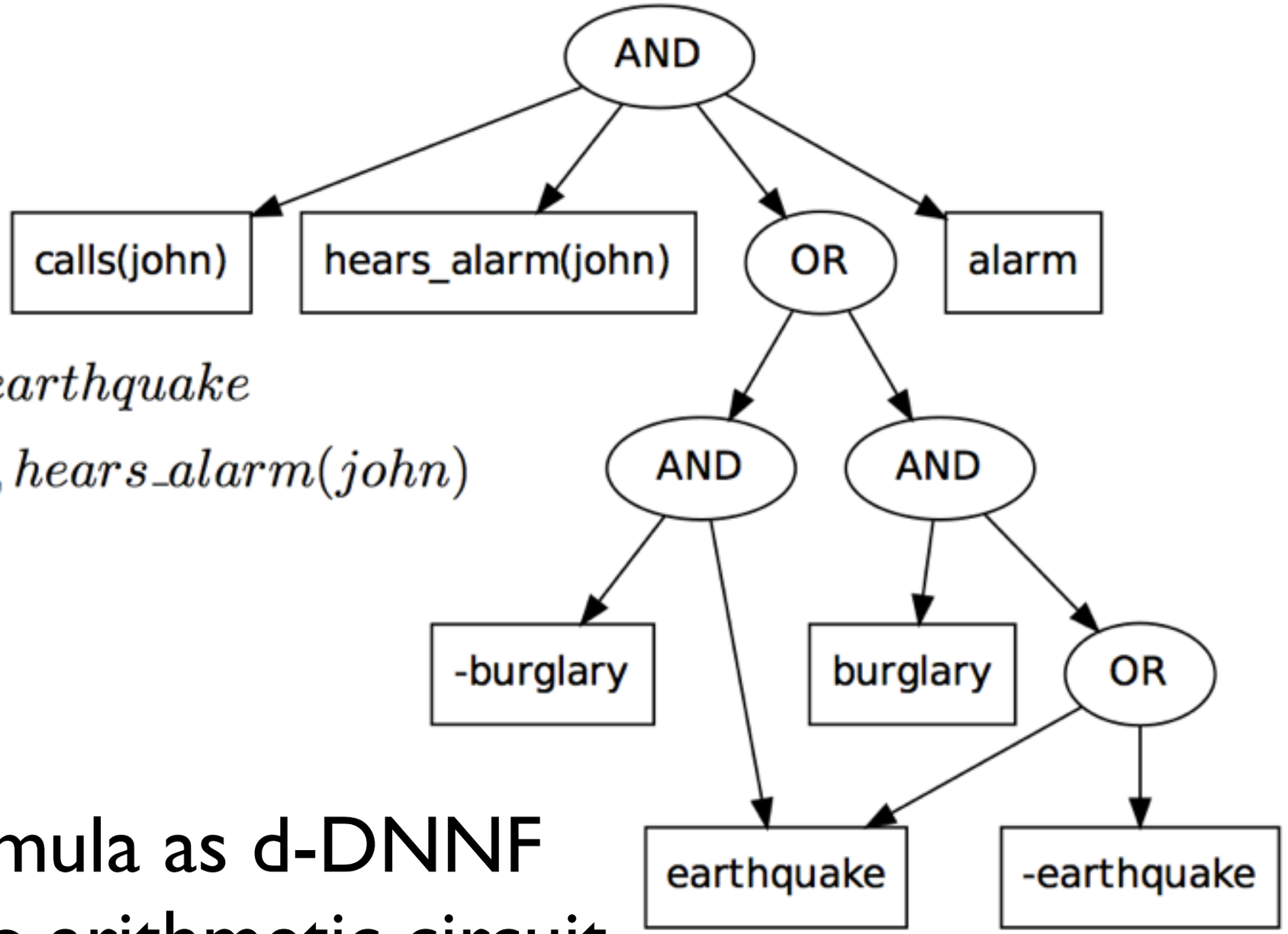
$$WMC((A \vee B) \wedge (\neg A \vee \neg B)) =$$
$$w(A) \times w(\neg B) + w(B) \times w(\neg B)$$

WMC using d-DNNFs

$alarm \leftrightarrow burglary \vee earthquake$

$calls(john) \leftrightarrow alarm, hears\_alarm(john)$

$calls(john)$



1. represent formula as d-DNNF
2. transform into arithmetic circuit
3. evaluate bottom-up

[Figure: Fierens et al, TPLP 14]

Knowledge Compilation

- Compile once - query many times …
- The knowledge compilation map (Darwiche and Marquis, JAIR 2002)
- Different representations, computational complexity of different operations, … (and, or, condition, ….)
- in StarAI — OBDDs, d-DNNFs, SDD (Darwiche), …
- Also state of the art for Bayesian net inference.

Example:
First-Order Model Counting

Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

models

$$\sum_{k=0}^{n} \binom{n}{k} 2^{n^2-k(n-k)}$$

Lifted Inference

Slide Guy Van den Broeck

Roadmap

1. Modeling
2. Inference
3. Learning
4. Applications

... with some detours on the way

Part III : Learning

Parameter Learning

e.g., webpage classification model

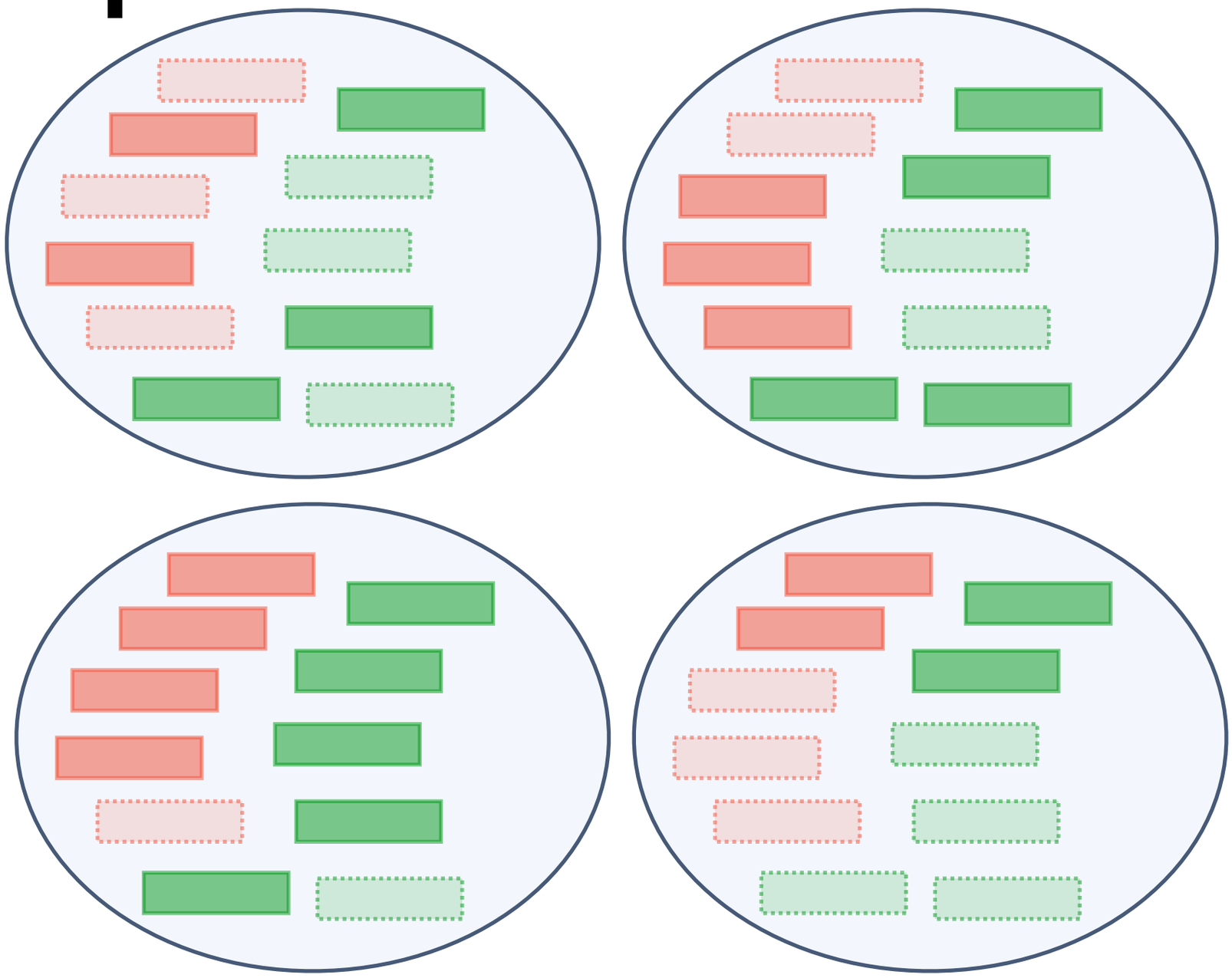
for each *CLASS1*, *CLASS2* and each *WORD*

?? :: link_class(Source,Target,*CLASS1*,*CLASS2*).
?? :: word_class(*WORD*,*CLASS*).

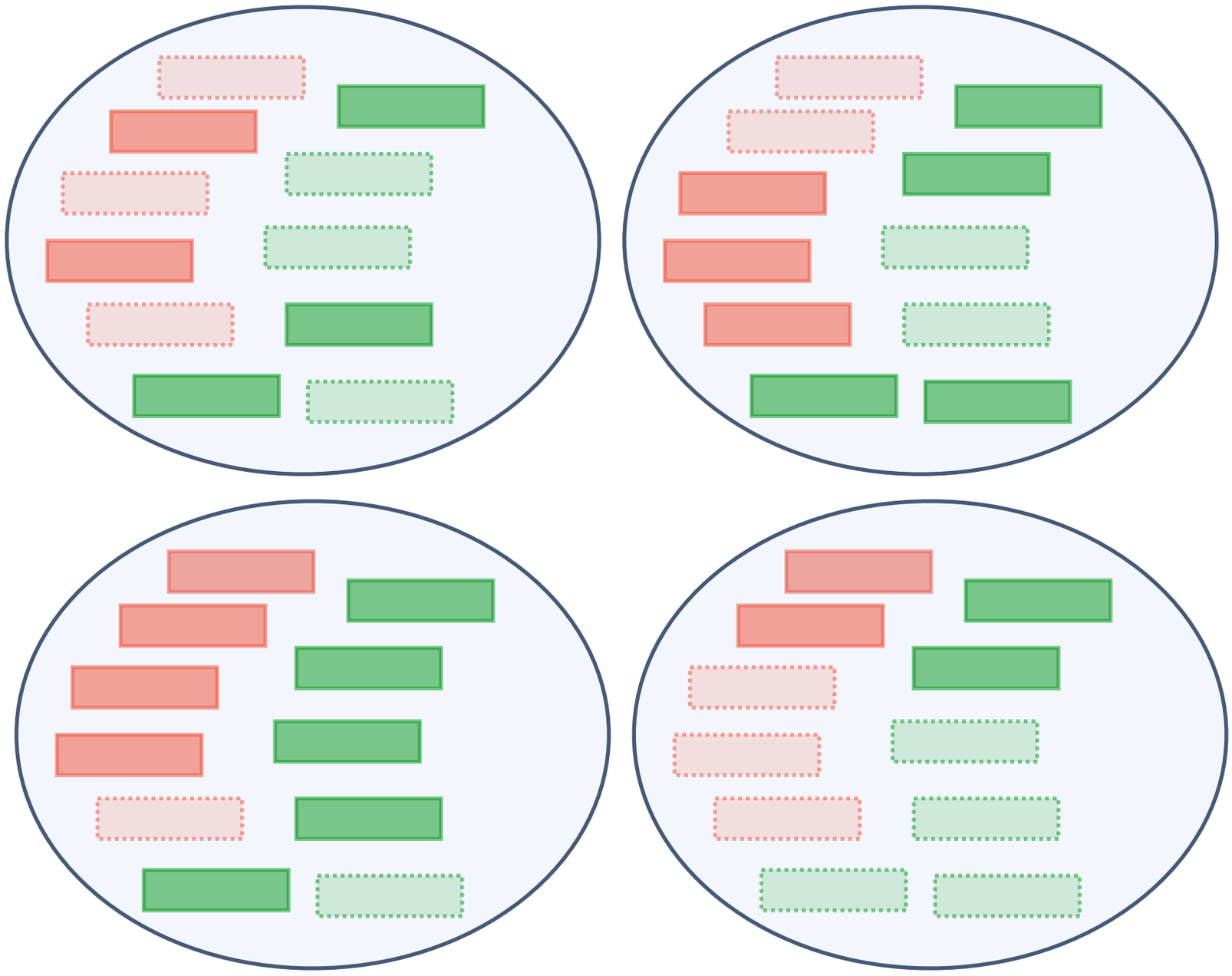
```
class(Page,C)    :-  has_word(Page,W), word_class(W,C).

class(Page,C)        :-  links_to(OtherPage,Page),
    class(OtherPage,OtherClass),
                         link_class(OtherPage,Page,OtherClass,C).
```

Sampling Interpretations

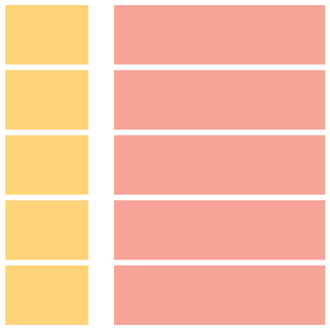
Sampling Interpretations

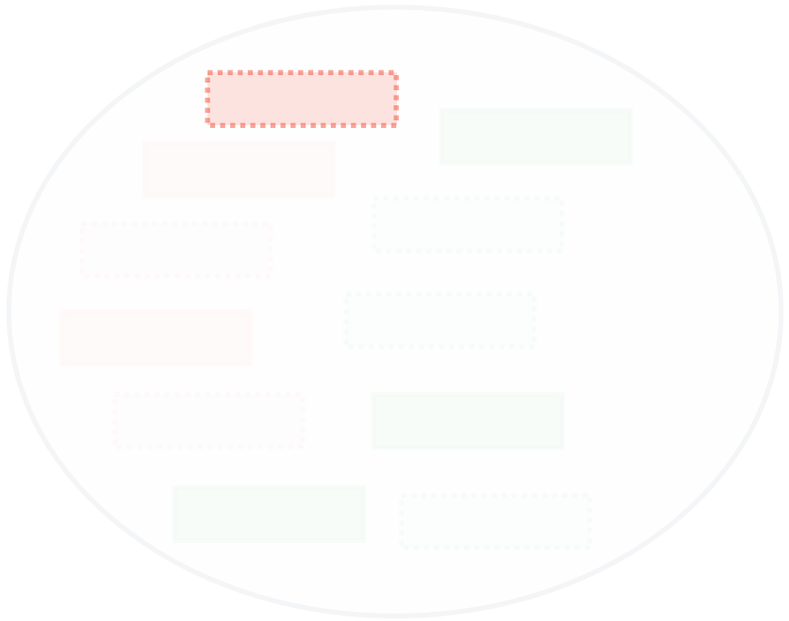


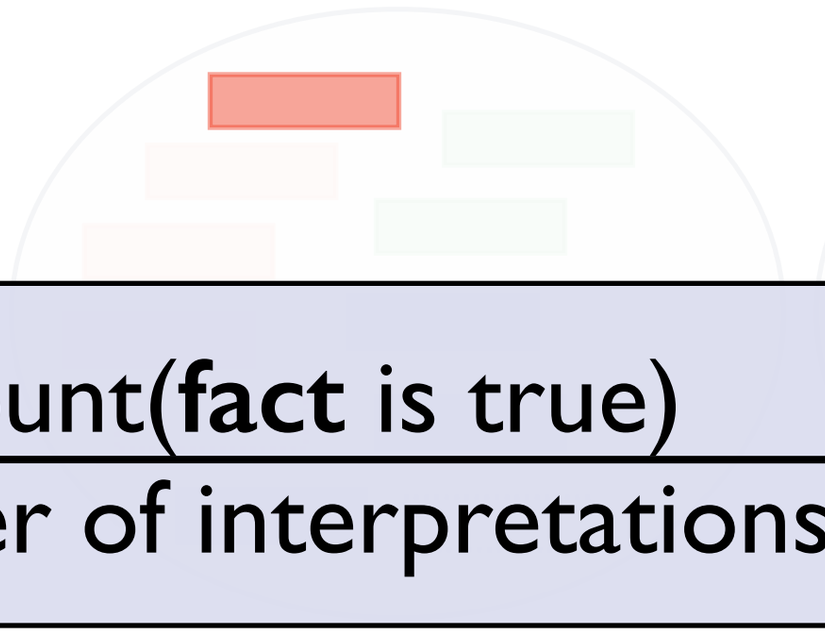
Parameter Estimation



Parameter Estimation

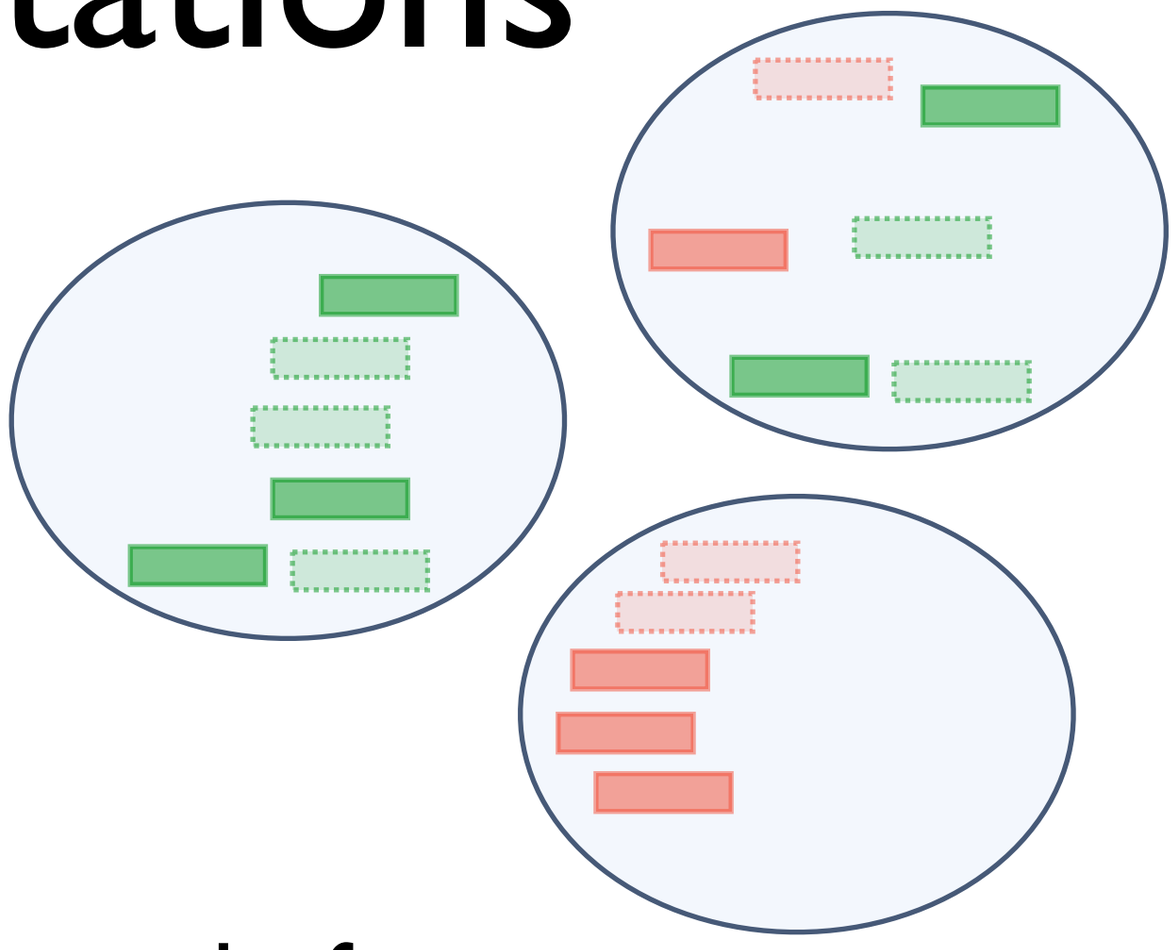






$$p(\textbf{fact}) = \frac{\text{count}(\textbf{fact} \text{ is true})}{\text{Number of interpretations}}$$

Learning from partial interpretations



- Not all facts observed
- Soft-EM
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

[Gutmann et al, ECML 11; Fierens et al, TPLP 14]

Bayesian Parameter Learning

- Learning as inference (e.g., Church)
- Prior distributions for parameters
- Given data, find most likely parameter values

Information Extraction in NELL

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/

Rule/Structure Learning

- Upgrade rule-learning to a probabilistic setting within a relational learning setting

- Exist for many of the formalisms (Markov Logic, ProbLog, etc)

- ProbFOIL works with a ***probabilistic*** logic program [De Raedt et al IJCAI 15]; it adapts Quinlan's FOIL

- Apply to probabilistic databases like NELL

Experiments

Table 4: Precision for different experimental setups and parameters (A: $m = 1, p = 0.99$, B: $m = 1000, p = 0.90$).

Setting	**athleteplaysforteam**		**athleteplayssport**		**teamplaysinleague**		**athleteplaysinleague**		**teamplaysagainstteam**	
train/test/rule	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
1: det/det/det	74.00	69.36	94.14	93.47	96.29	82.15	80.95	74.14	73.40	73.86
2: det/prob/det	73.51	69.57	97.53	94.85	96.70	87.83	90.83	77.73	73.70	73.35
3: det/prob/prob	74.67	69.82	95.86	94.74	96.35	82.57	82.26	75.29	73.84	74.34
4: det/prob/prob	77.25	73.87	96.53	96.04	98.00	90.59	84.91	79.36	77.26	77.83
5: det/prob/prob	74.76	69.97	95.85	94.69	96.44	82.51	81.99	75.07	73.90	74.16
6: prob/prob/det	75.83	73.11	93.40	93.76	94.44	93.67	79.41	79.42	80.87	80.60
7: prob/prob/prob	78.31	73.72	95.62	95.10	98.84	91.86	96.94	79.49	85.78	81.81

Table 3: Learned relational rules for the different predicates (fold 1).

0.9375::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B)	←	athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B)	←	athleteplaysinleague(A,V1), teamplaysinleague(B,V1).
0.9070::athleteplayssport(A,B)	←	athleteledsportsteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B),teamalsoknownas(V1,V2).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V1), teamplayssport(V1,B).
0.9286::athleteplaysinleague(A,B)	←	athleteledsportsteam(A,V1), teamplaysinleague(V1,B).
0.7868::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplaysinleague(V1,B).
0.9384::athleteplaysinleague(A,B)	←	athleteplayssport(A,V2), athleteplayssport(V1,V2), teamplaysinleague(V1,B).
0.9024::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V1), teamplaysinleague(V1,B).

Roadmap

1. Modeling
2. Inference
3. Learning
4. Applications

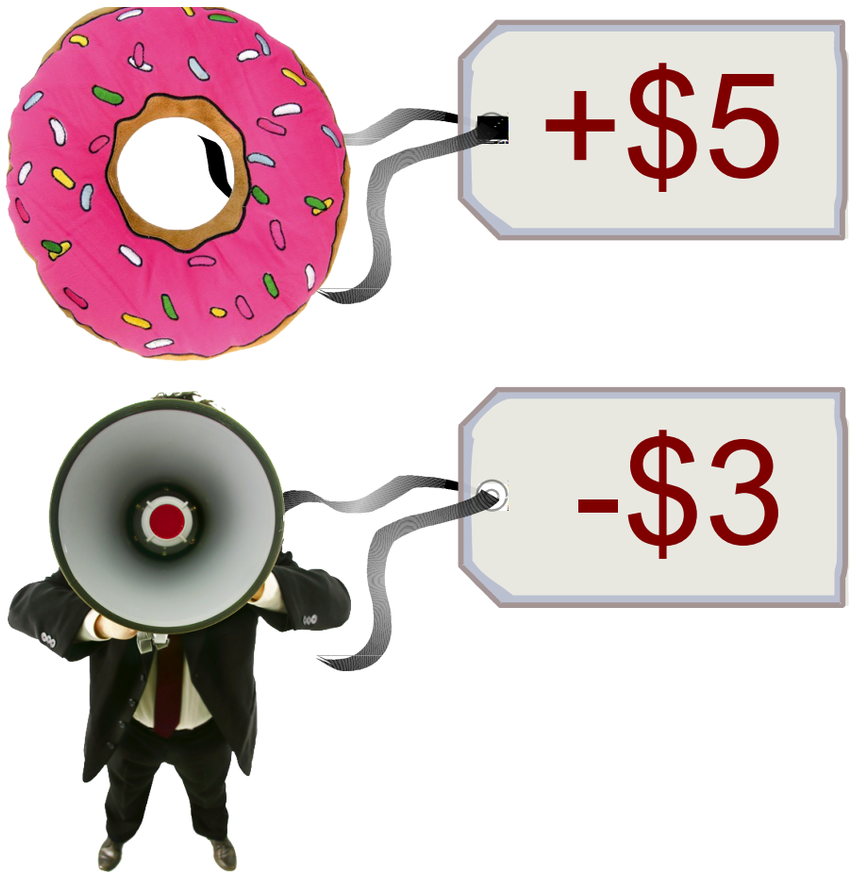
... with some detours on the way

Part IV : Applications

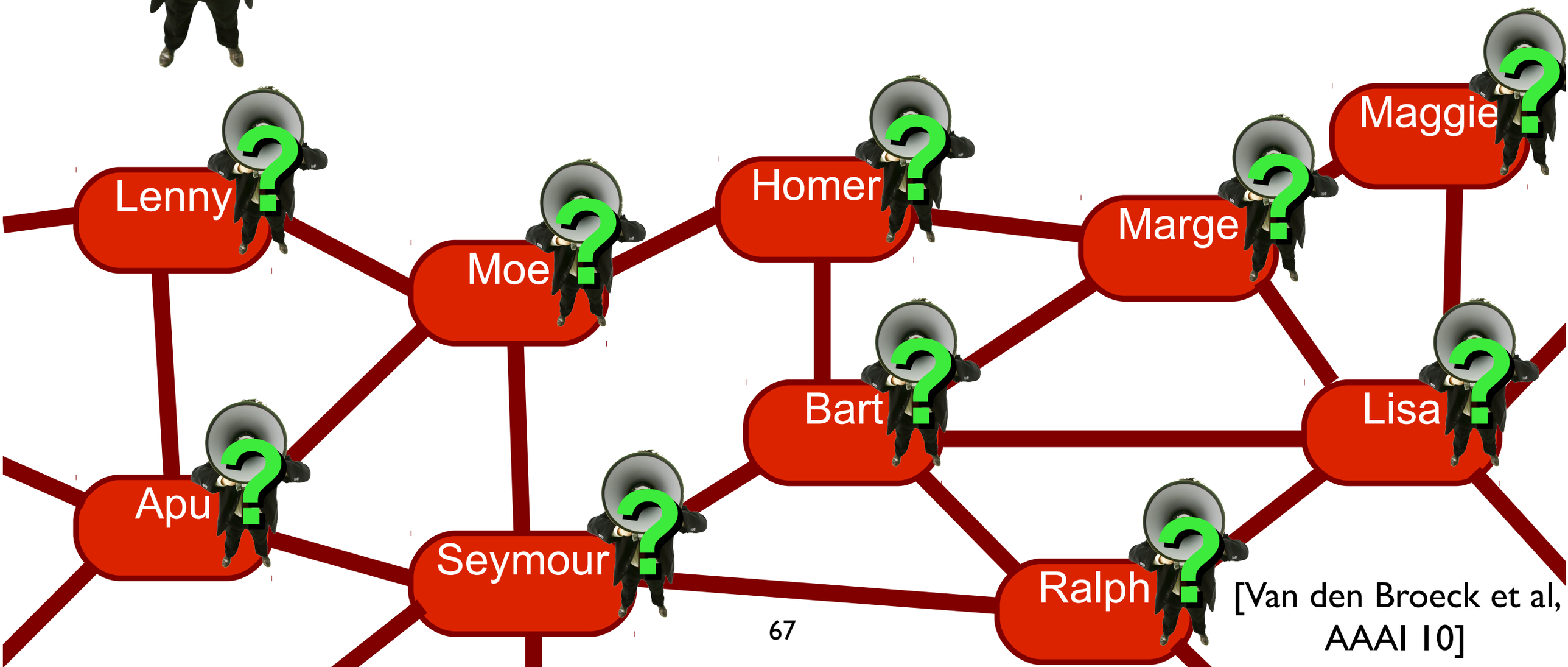
Biological Network Analysis

Decision Theoretic Extension

Viral Marketing

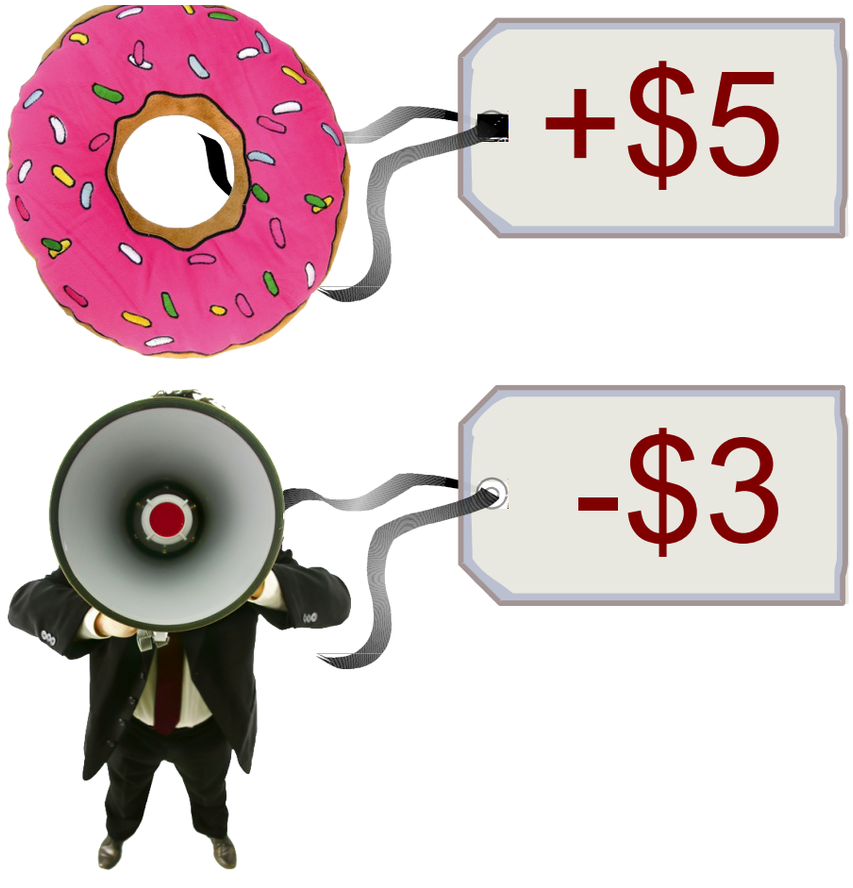


Which advertising strategy maximizes expected profit?

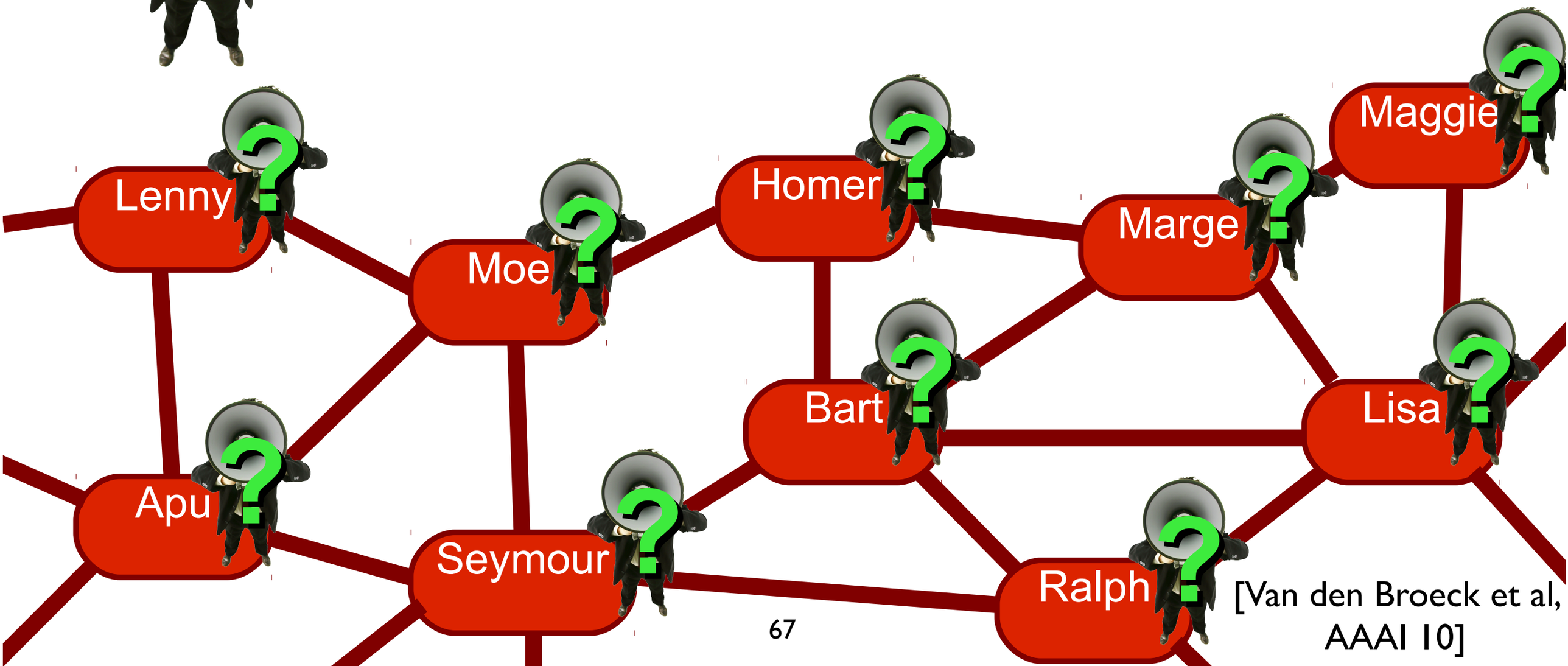


[Van den Broeck et al, AAAI 10]

Viral Marketing

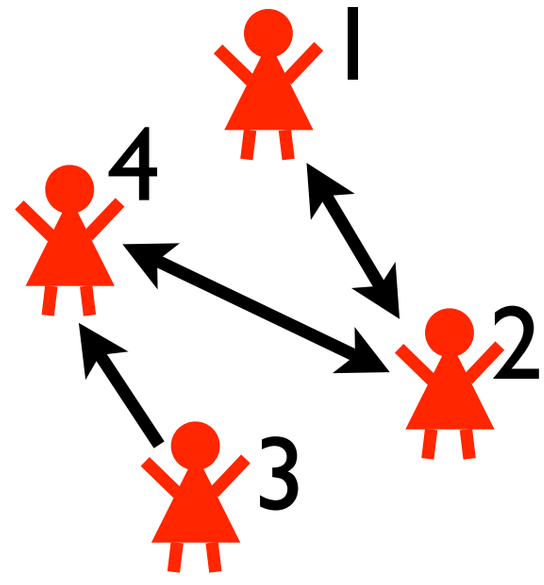


decide truth values of some atoms



[Van den Broeck et al, AAAI 10]

DTProbLog



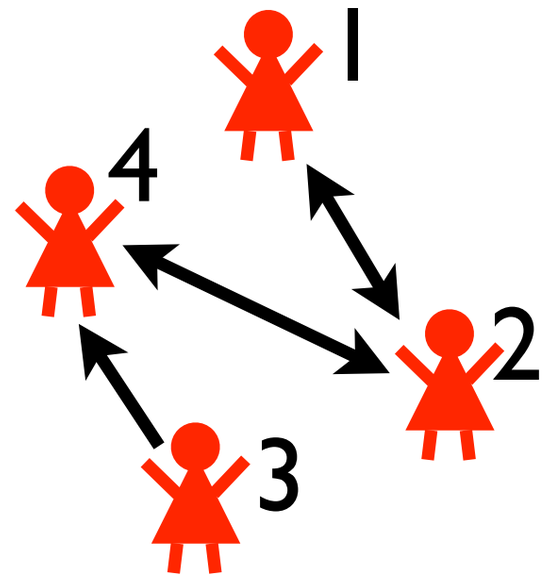

```
person(1).
person(2).
person(3).
person(4).
```

```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

```
? :: marketed(P) :- person(P).
```

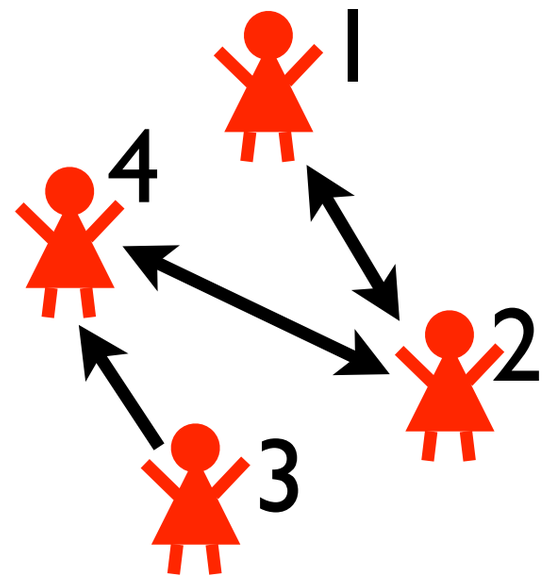
decision fact: true or false?




```
person(1).
person(2).
person(3).
person(4).
```

```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

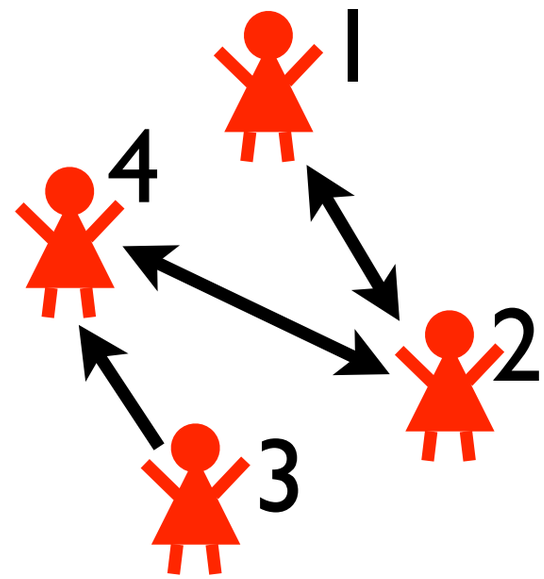



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).
```

probabilistic facts
+ logical rules

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

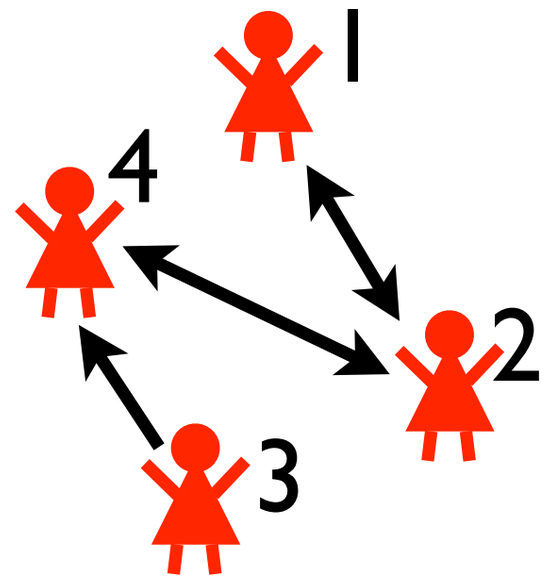



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

utility facts: cost/reward if true

DTProbLog

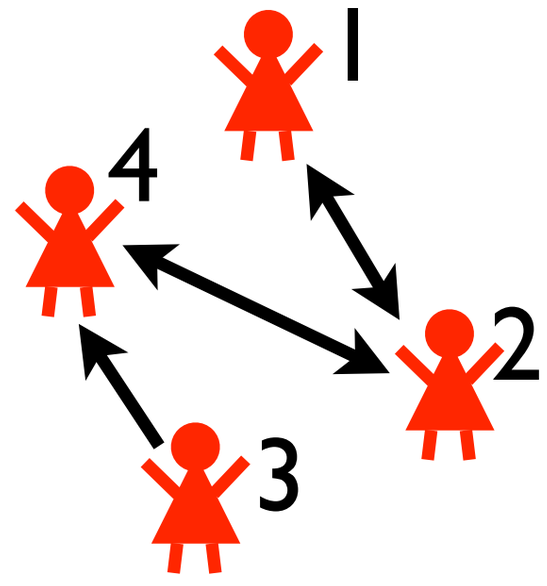



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

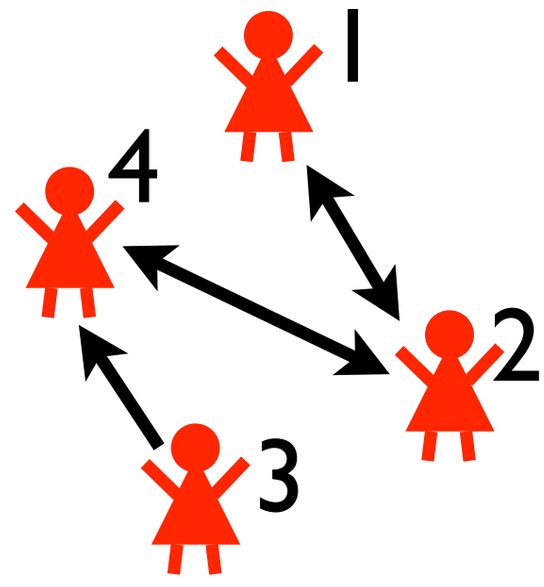
DTProbLog

```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```




```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

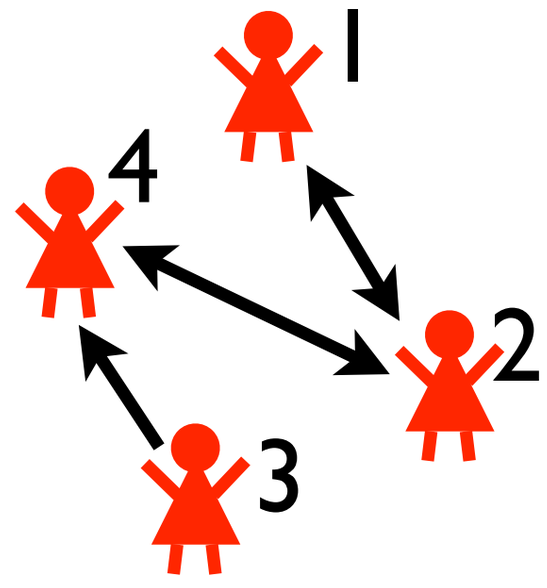



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)
```

DTProbLog

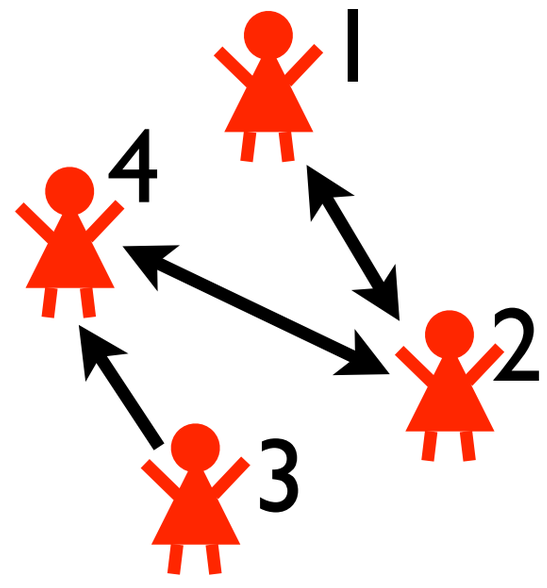



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)
```

DTProbLog

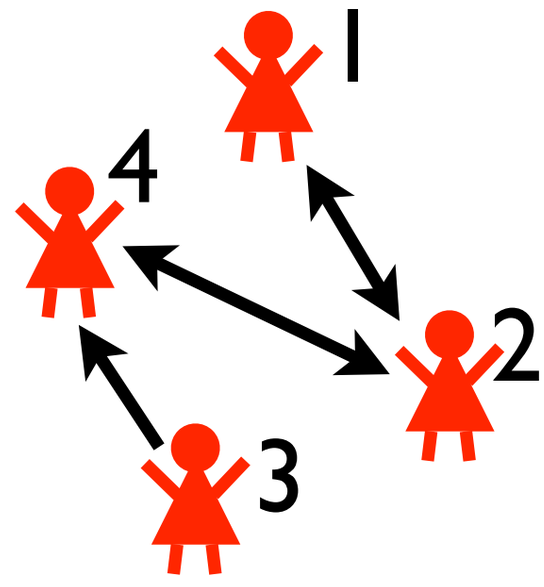



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)

buys(1)    buys(2)
```

DTProbLog



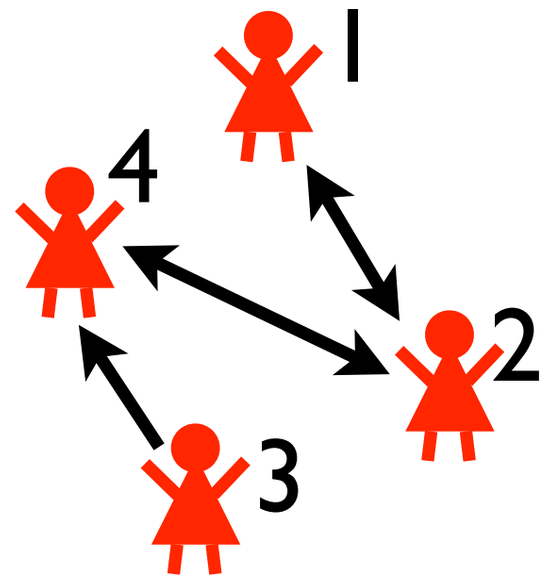

```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

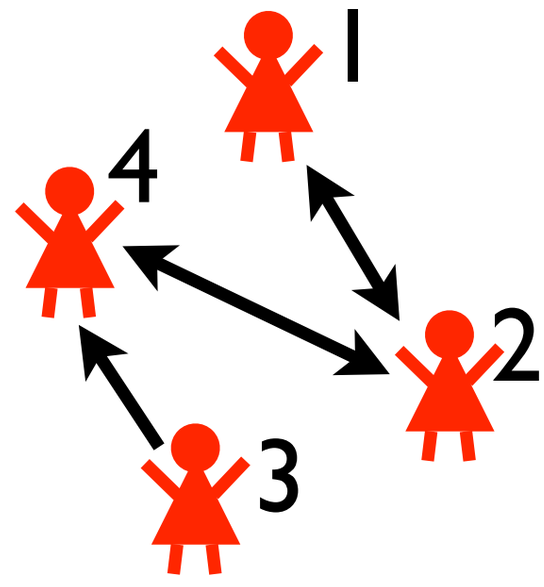
utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)

buys(1)    buys(2)
```

world contributes 0.0032×4 to expected utility of strategy

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

task: find strategy that maximizes expected utility
solution: using ProbLog technology

Phenetic

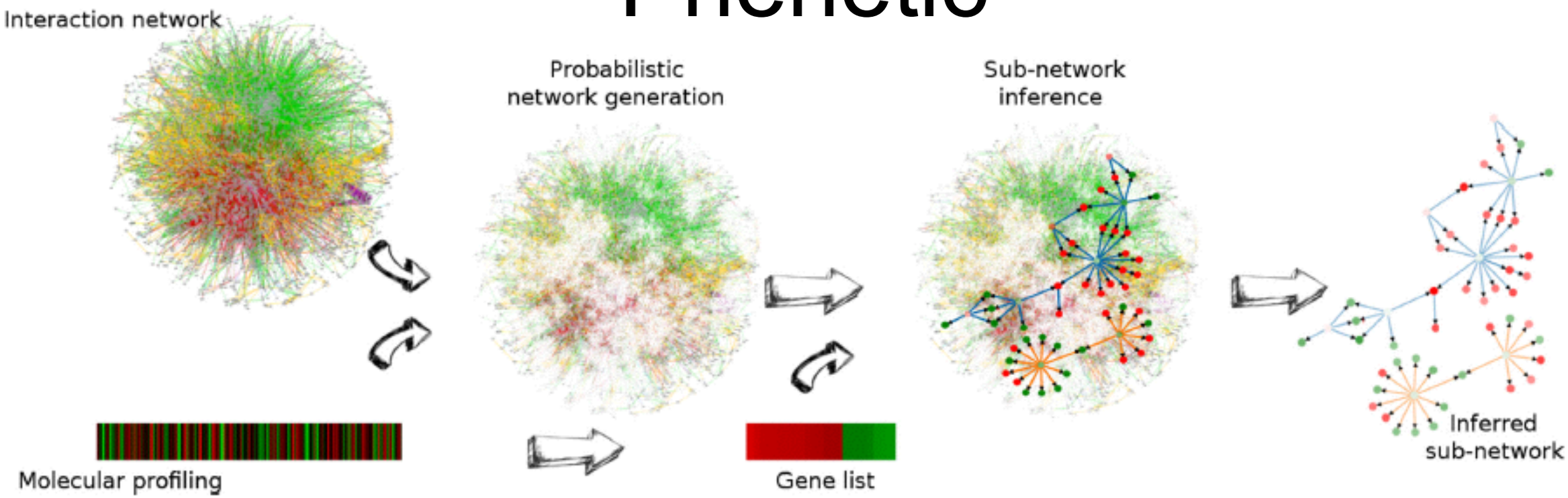


Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for a wide variety of organisms are readily available from the web server. Using the uploaded user-generated molecular data the interaction network is converted into a probabilistic network: edges receive a probability proportional to the levels measured for the terminal nodes in the molecular profiling data set. This probabilistic interaction network is used to infer the sub-network that best links the genes from the gene list. The inferred sub-network provides a trade-off between linking as many genes as possible from the gene list and selecting the least number of edges.

- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

- Interaction network:
  - 3063 nodes
    - Genes
    - Proteins
  - 16794 edges
    - Molecular interactions
    - Uncertain

- Goal: connect causes to effects through common subnetwork
  - = Find mechanism
- Techniques:
  - DTProbLog
  - Approximate inference

[De Maeyer et al., Molecular Biosystems 13, NAR 15]

Answering Probabilistic Exam Questions

A solver based on ProbLog

Our goal

Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

Our goal

Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

The answer is 0.235941.

Our goal

Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

The answer is 0.235941.

combination of mathematics
and natural language processing
[Dries et al. IJCAI 2017]

Inspiration

Aristo

Growing thicker fur in the winter helps some animals to

(A) hide from danger
(B) attract a mate
(C) find food
(D) keep warm

ARISTO ANSWERED:

▸ (D) keep warm — Explain

OTHER ANSWERS:

▸ (C) find food — Explain

some examples

ALLEN INSTITUTE *for* ARTIFICIAL INTELLIGENCE

Inspiration

Aristo

Growing thicker fur in the winter helps some animals to

(A) hide from danger
(B) attract a mate
(C) find food
(D) keep warm

ARI

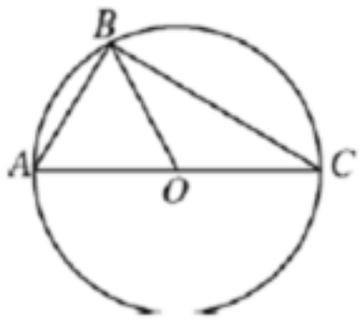
▸ (D) keep warm

OT

▸ (C) find food

some examples

In the figure above, triangle ABC is inscribed in the circle with center O and diameter AC. If AB=AO, what is the degree measure of angle ABO?



(A) 15°
(B) 30°
(C) 45°
(D) 60°
(E) 90°

Euclid / Geos

ALLEN INSTITUTE
for ARTIFICIAL INTELLIGENCE

Our approach

Our approach

natural language

Our approach

natural language

off-the-shelf NLP tools
+ rule-based system

Our approach

natural language

off-the-shelf NLP tools
+ rule-based system

specification language

Our approach

natural language

off-the-shelf NLP tools
+ rule-based system

specification language

solver

Our approach

natural language

off-the-shelf NLP tools
+ rule-based system

specification language

solver

solution

Example

Mike has a bag with 4 white, 8 blue, and 6 red marbles.

He takes one marble from the bag and it is red.

What is the probability that the second marble he takes from the bag is white?

Example

Mike has a bag with 4 white, 8 blue, and 6 red marbles.

He takes one marble from the bag and it is red.

What is the probability that the second marble he takes from the bag is white?

setup

multiset bag

Values(color) = {white, blue, red}

#white(bag) = 4
#blue(bag) = 8
#red(bag) = 6

Example

Mike has a bag with 4 white,
8 blue, and 6 red marbles.

multiset bag

Values(color) = {white, blue, red}

#white(bag) = 4
#blue(bag) = 8
#red(bag) = 6

He takes one marble from the bag and it is red.

action + observation

first = take(bag)
#first = 1
rest(first) = bag \ first
observe #red(first) = #first

What is the probability that the second marble he takes from the bag is white?

Example

Mike has a bag with 4 white, 8 blue, and 6 red marbles.

multiset bag

Values(color) = {white, blue, red}

#white(bag) = 4
#blue(bag) = 8
#red(bag) = 6

He takes one marble from the bag and it is red.

first = take(bag)
#first = 1
rest(first) = bag \ first

What is the probability that the second marble he takes from the bag is white?

action + question

snd = take(rest(first))
rest(snd) = rest(first) \ snd
#snd = 1
probability #white(snd) = #snd

Example

Mike has a bag with 4 white, 8 blue, and 6 red marbles.

He takes one marble from the bag and it is red.

What is the probability that the second marble he takes from the bag is white?

multiset bag

Values(color) = {white, blue, red}

#white(bag) = 4
#blue(bag) = 8
#red(bag) = 6

first = take(bag)
#first = 1
rest(first) = bag \ first

observe #red(first) = #first

snd = take(rest(first))
rest(snd) = rest(first) \ snd
#snd = 1
probability #white(snd) = #snd

Examples

Other types of questions can be represented in the same formalism

A die is thrown 3 times.
Find the probability that the sum of the dots is at least 5.

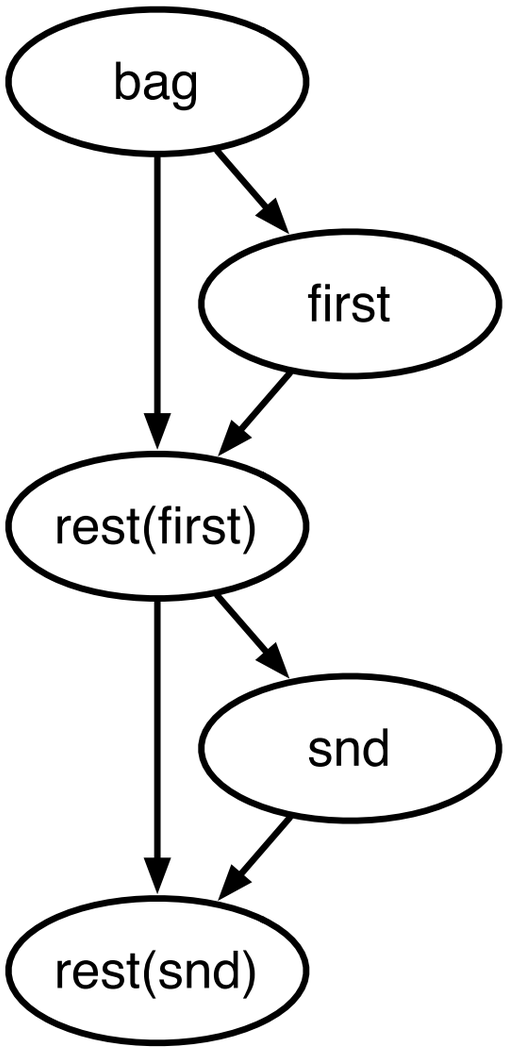
A blood disease is found in 2 percent of the persons in a certain population. A new blood test will correctly identify 96 percent of the persons with the disease and 94 percent of the persons without the disease. What is the probability that a person who is called negative by the blood test actually does not have the disease?

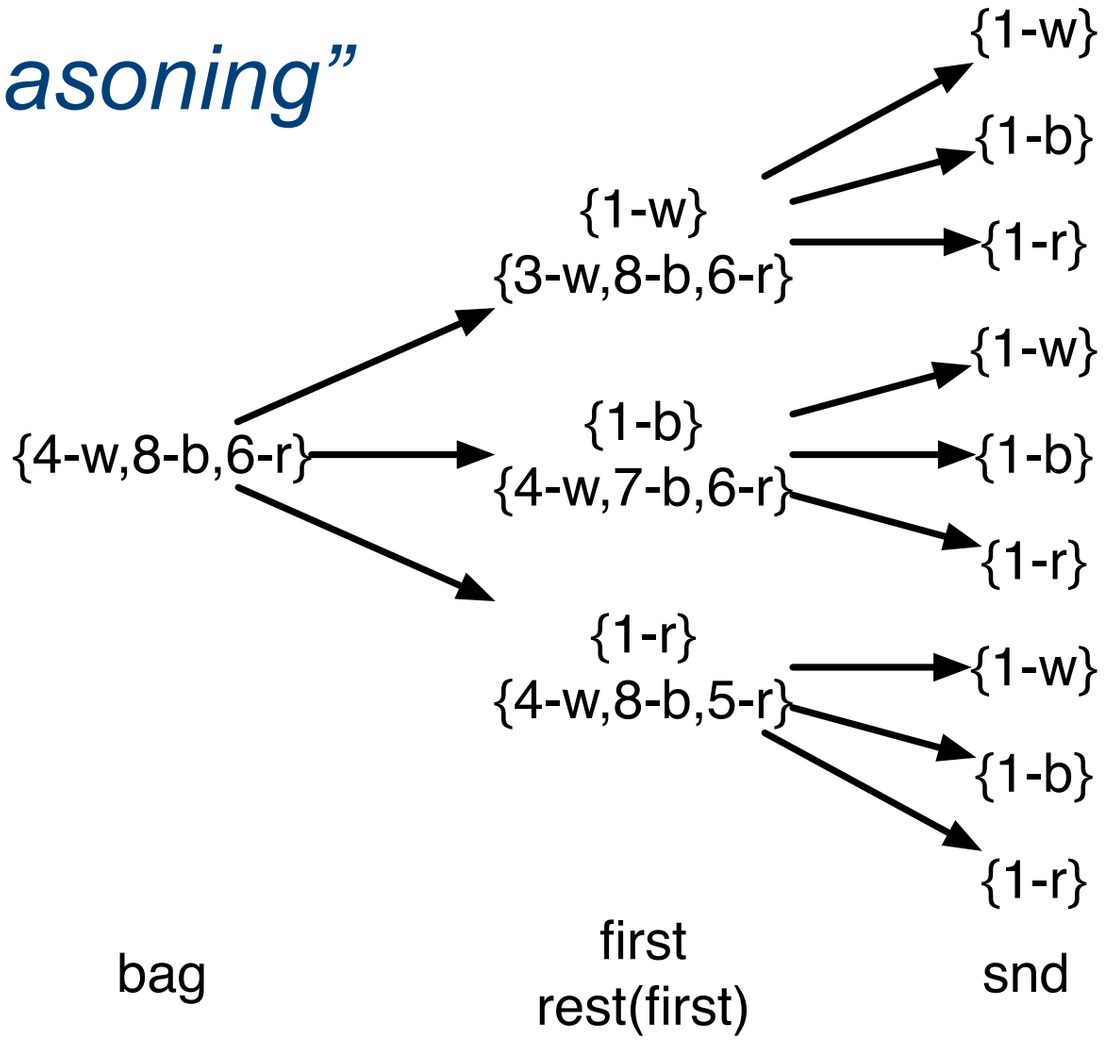
Solver

take into account exchangeability

all white balls are interchangeable

“lifted reasoning”





we use also a constraint solver to determine actual numbers

Solver

takes advantage of underlying solver technology

```
4/18::white(1); 8/18::blue(1); 6/18::red(1).

3/17::white(2); 8/17::blue(2); 6/17::red(2) :- white(1).
4/17::white(2); 7/17::blue(2); 6/17::red(2) :- blue(1).
4/17::white(2); 8/17::blue(2); 5/17::red(2) :- red(1).

evidence(red(1)).
query(white(2)).
```

Solver

takes advantage of underlying solver technology

```
4/18::white(1); 8/18::blue(1); 6/18::red(1).

3/17::white(2); 8/17::blue(2); 6/17::red(2) :- white(1).
4/17::white(2); 7/17::blue(2); 6/17::red(2) :- blue(1).
4/17::white(2); 8/17::blue(2); 5/17::red(2) :- red(1).

evidence(red(1)).
query(white(2)).
```

```
6/18::red(1).

4/17::white(2) :- red(1).

evidence(red(1)).
query(white(2)).
```

Data

2376 questions on probability

structure of the problems labeled and solved by hand

groups of objects

probabilistic events

Results

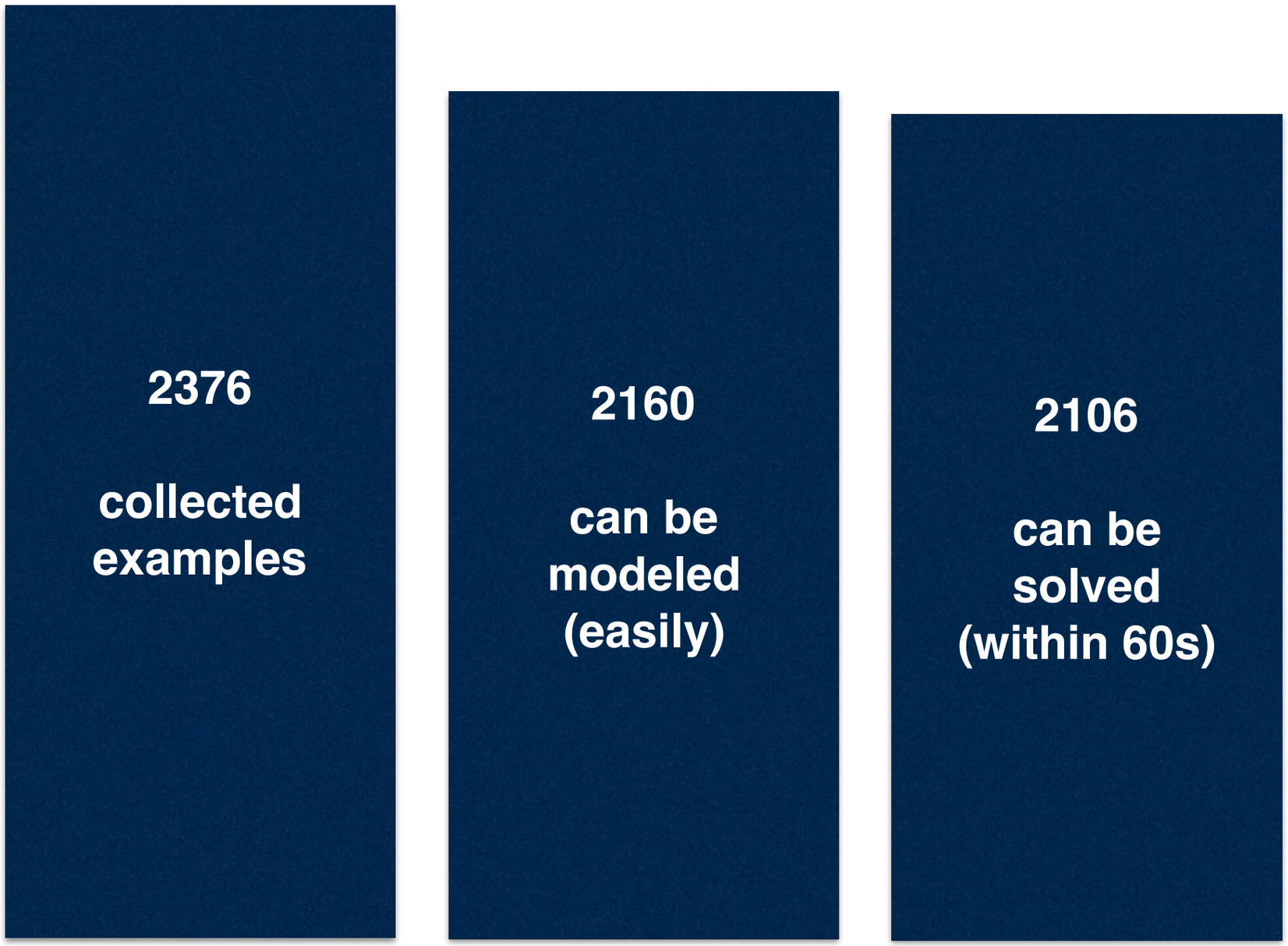
Results

2376

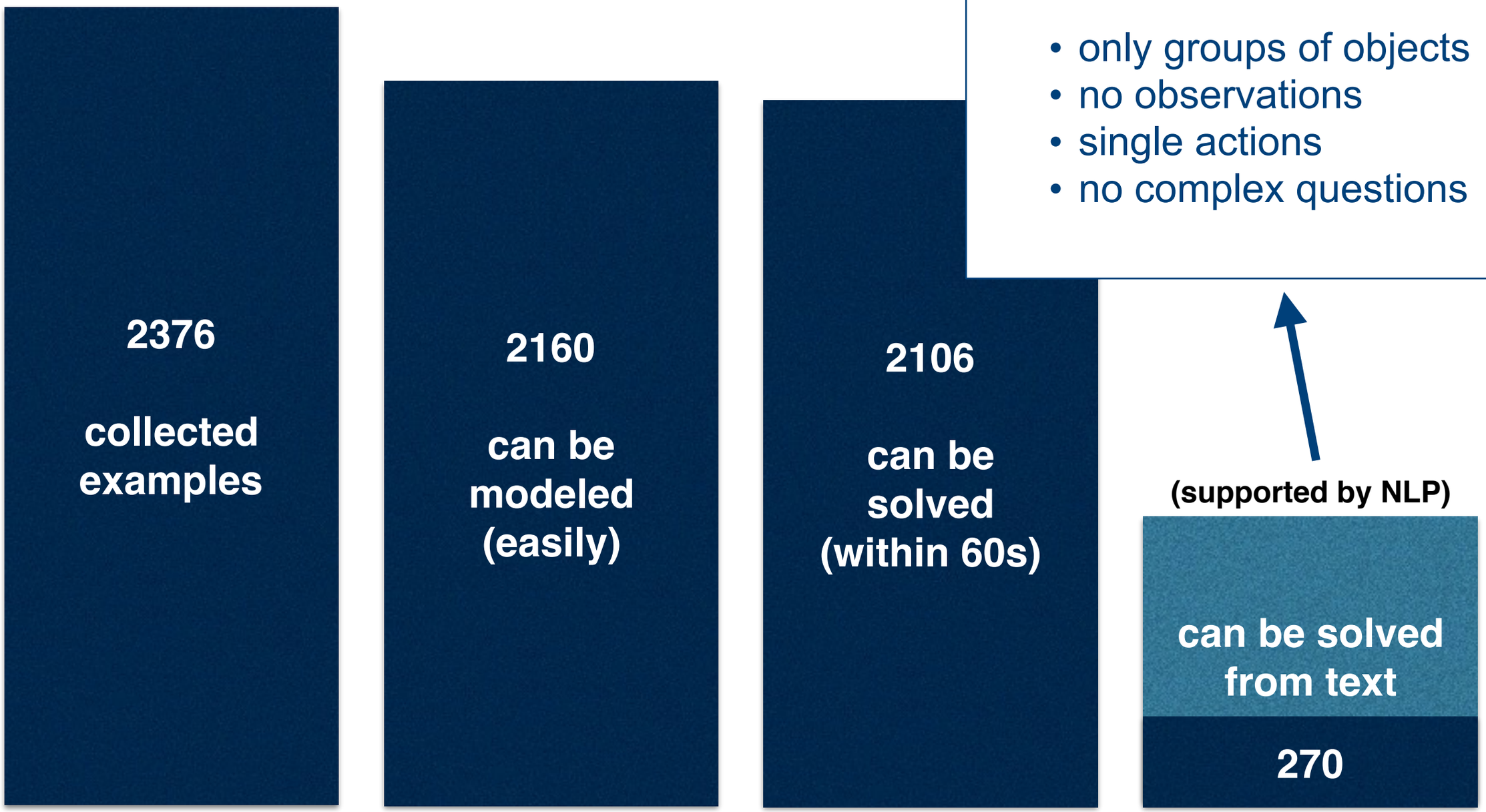
collected examples

Results

Results


2376
collected examples
2160
can be modeled (easily)
2106
can be solved (within 60s)

Results

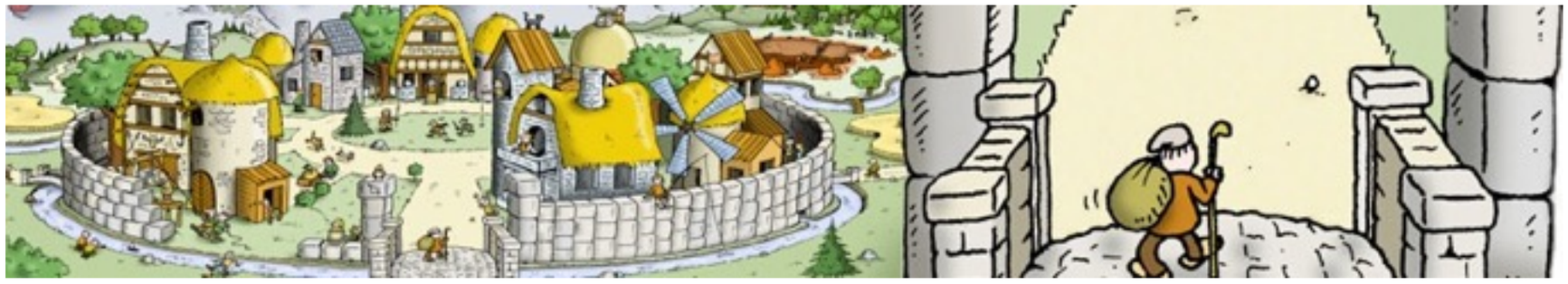


Limitations:

- only groups of objects
- no observations
- single actions
- no complex questions

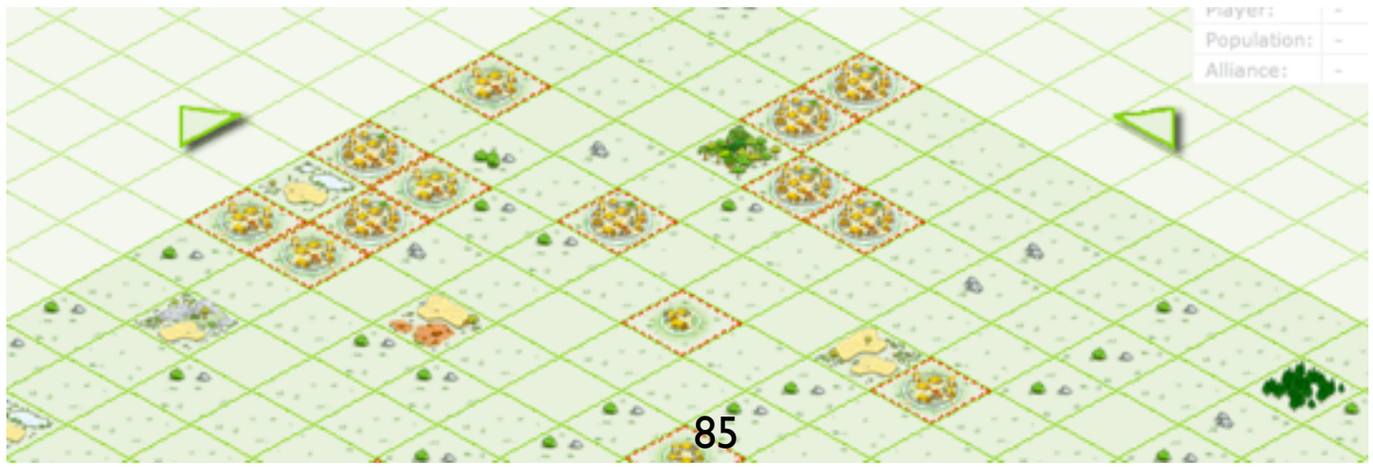
Dynamic worlds

Dynamics: Evolving Networks



- *Travian*: A massively multiplayer real-time strategy game
- Commercial game run by TravianGames GmbH
- ~3.000.000 players spread over different "worlds"
- ~25.000 players in one world

[Thon et al. ECML 08]



World Dynamics

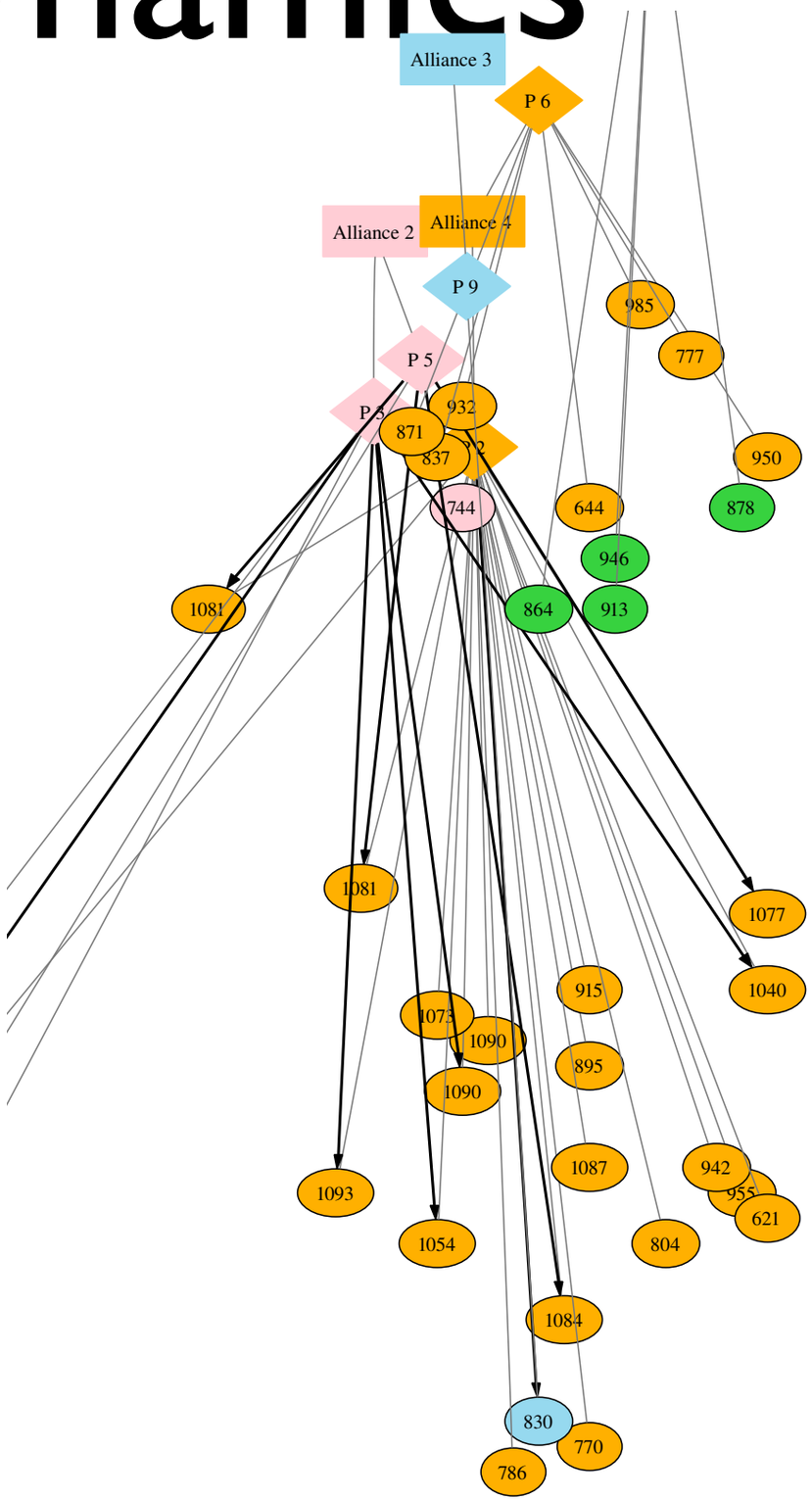
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

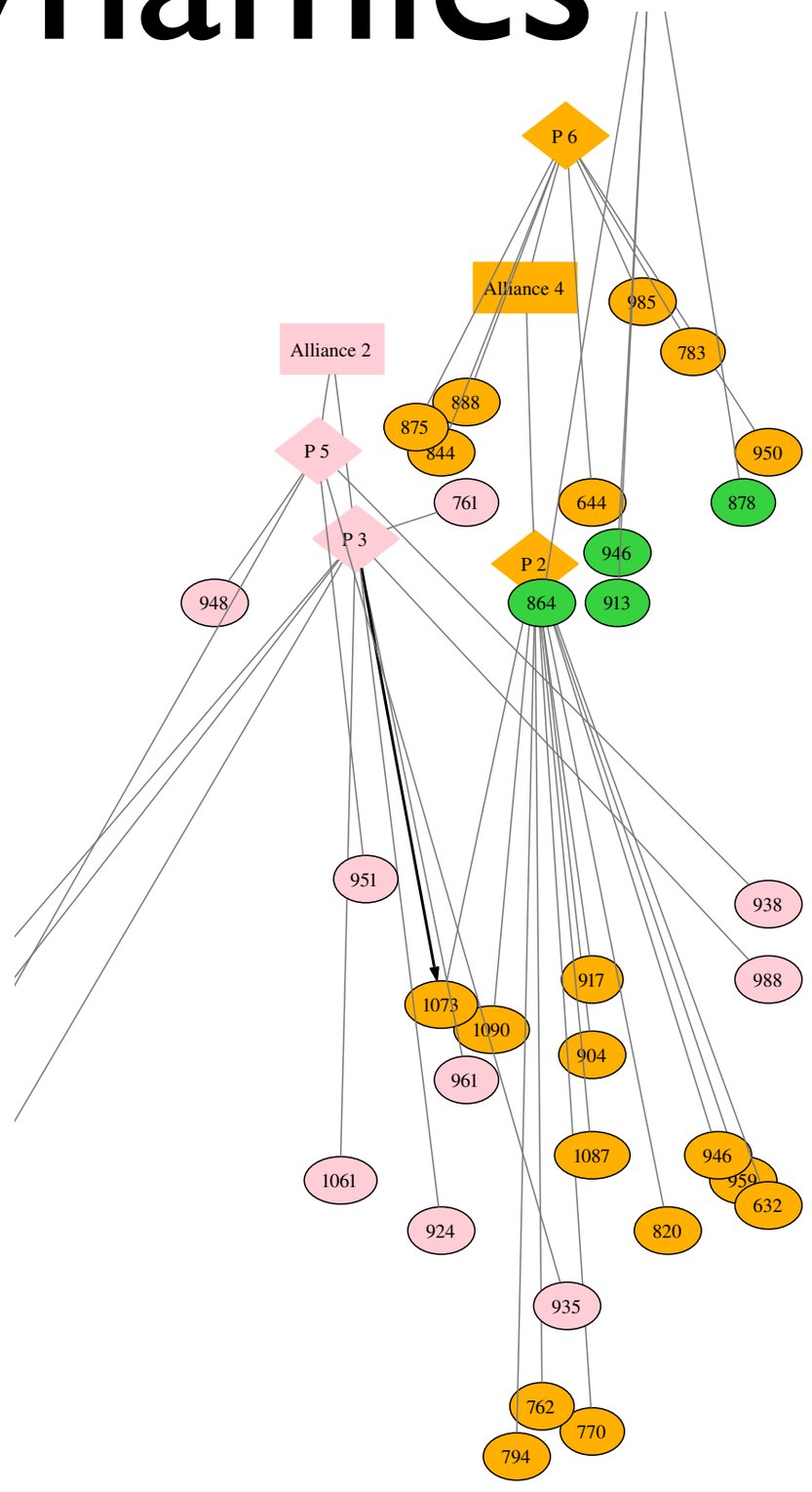
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

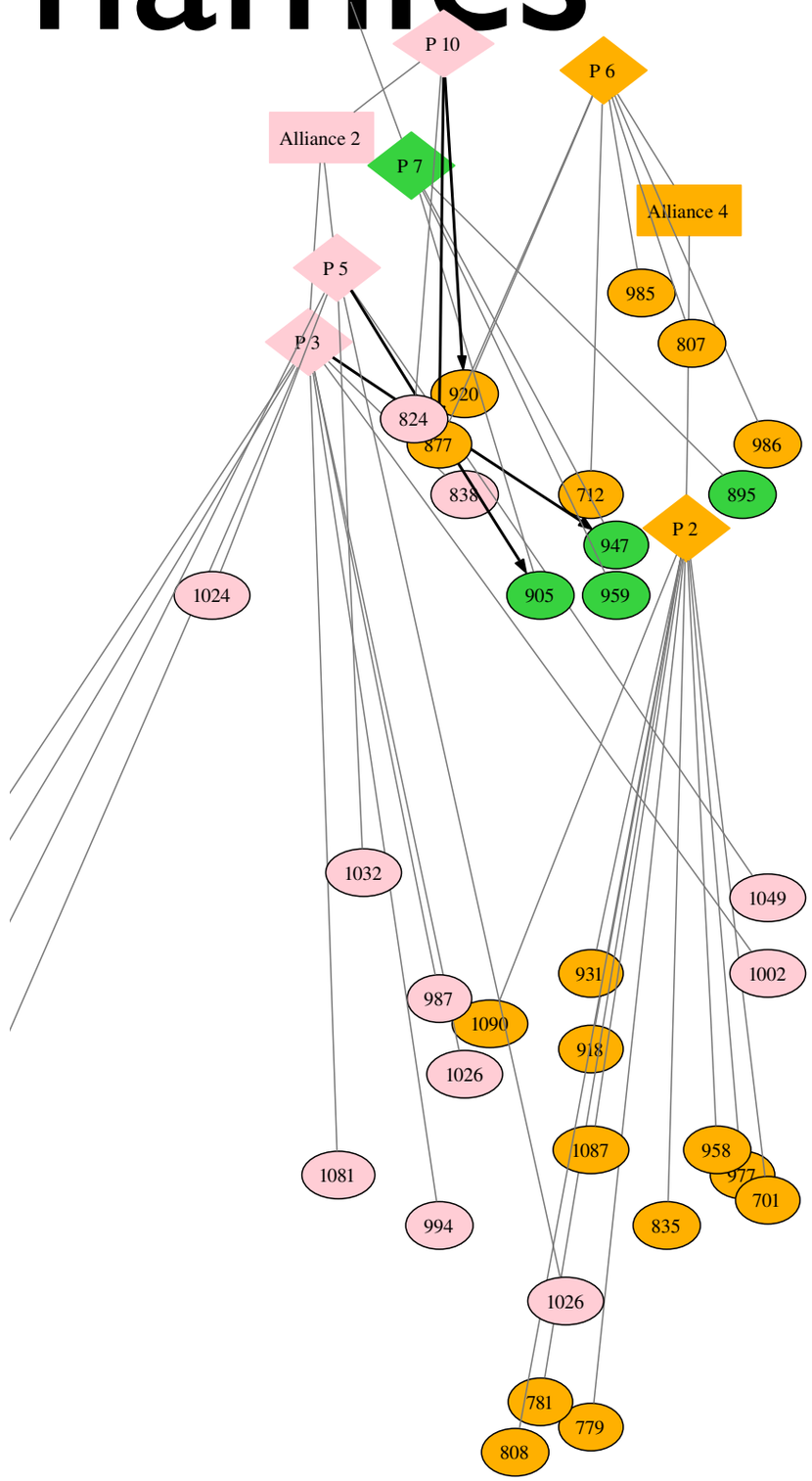
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

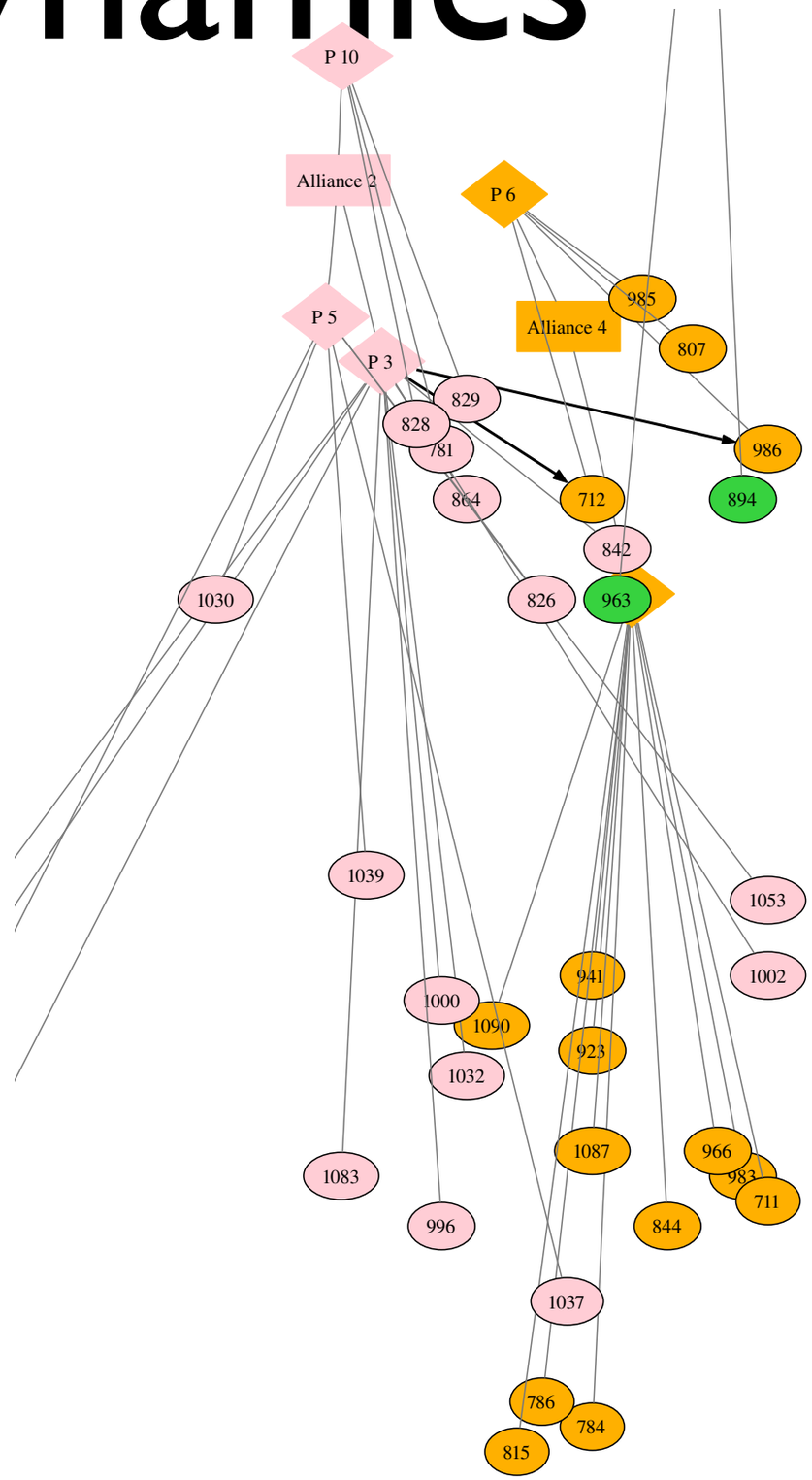
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

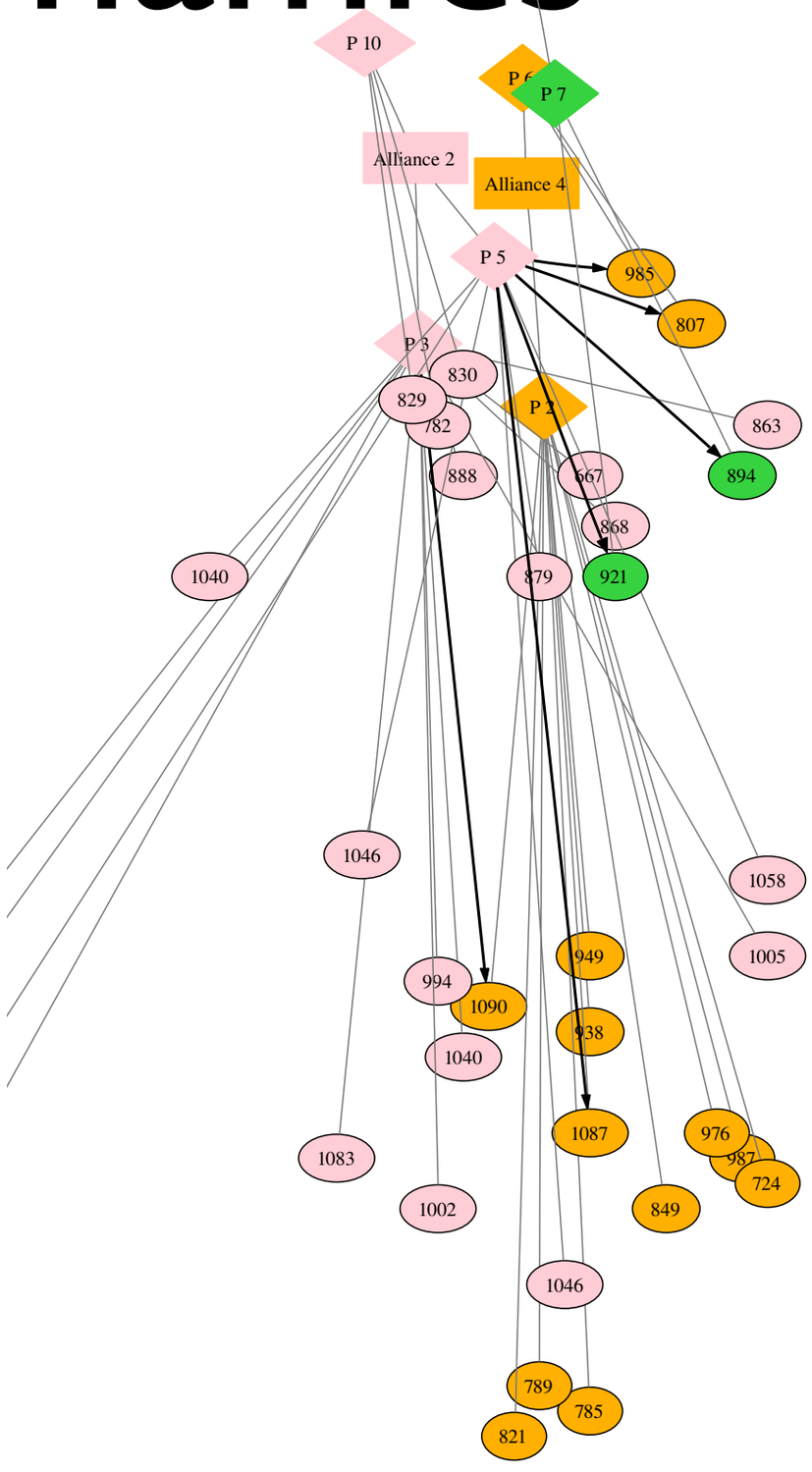
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

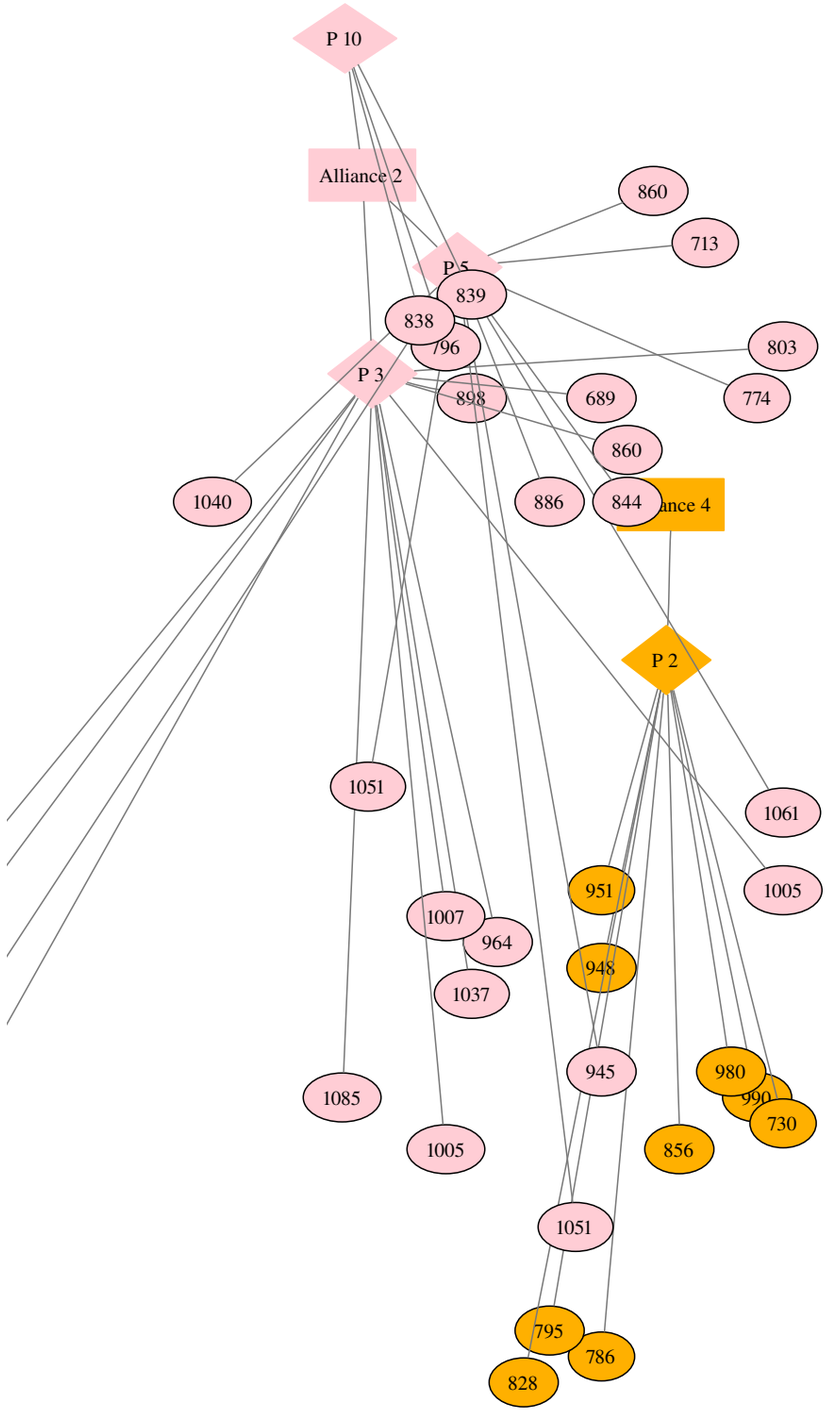
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]

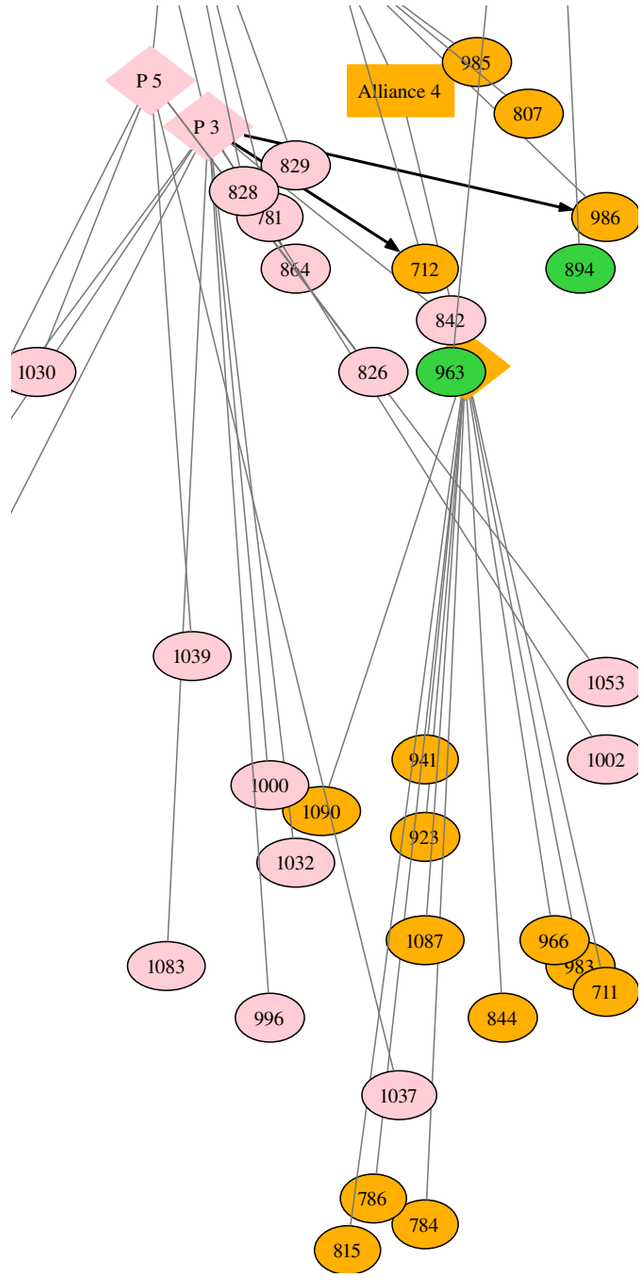


CPT-Rules

$$\frac{b_1, \ldots b_n}{\text{cause}} \rightarrow \frac{h_1 : p_1 \vee \ldots \vee h_m : p_m}{\text{effect}}$$

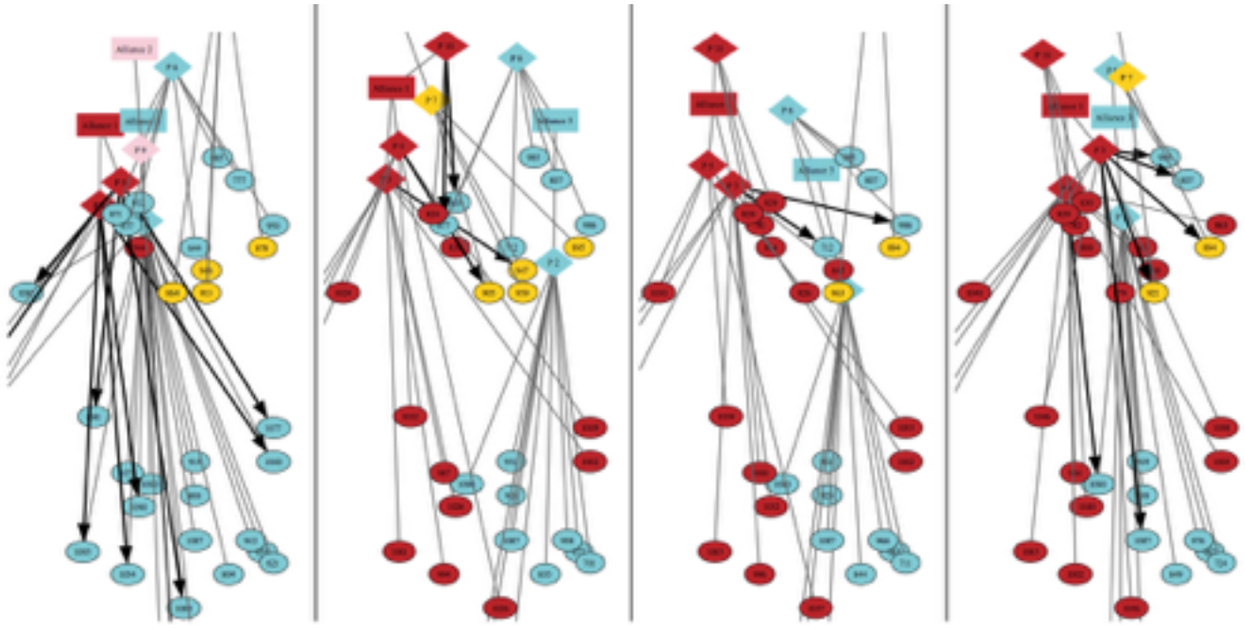
$$city(C, Owner), city(C2, Attacker), close(C, C2) \rightarrow$$
$$conquest(Attacker, C2) \ : \ p \ \vee \ nil \ : \ (1-p)$$

conquer a city which is close
P(conquest(),Time+5) ?
learn parameters



Thon et al. MLJ 11

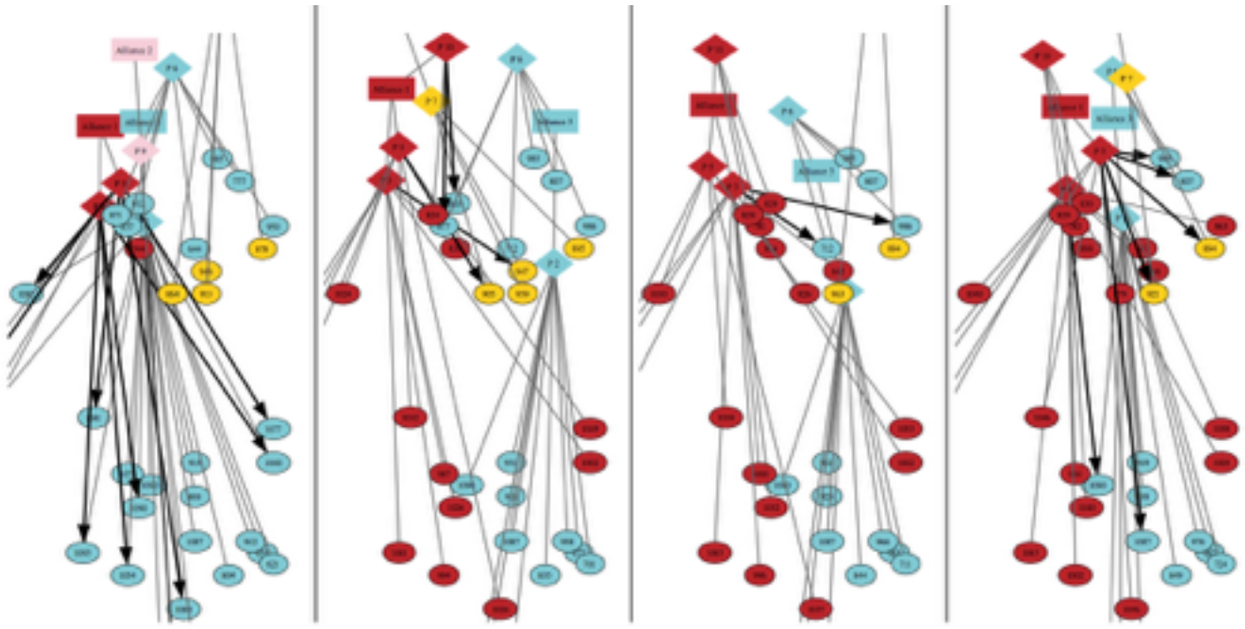
Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



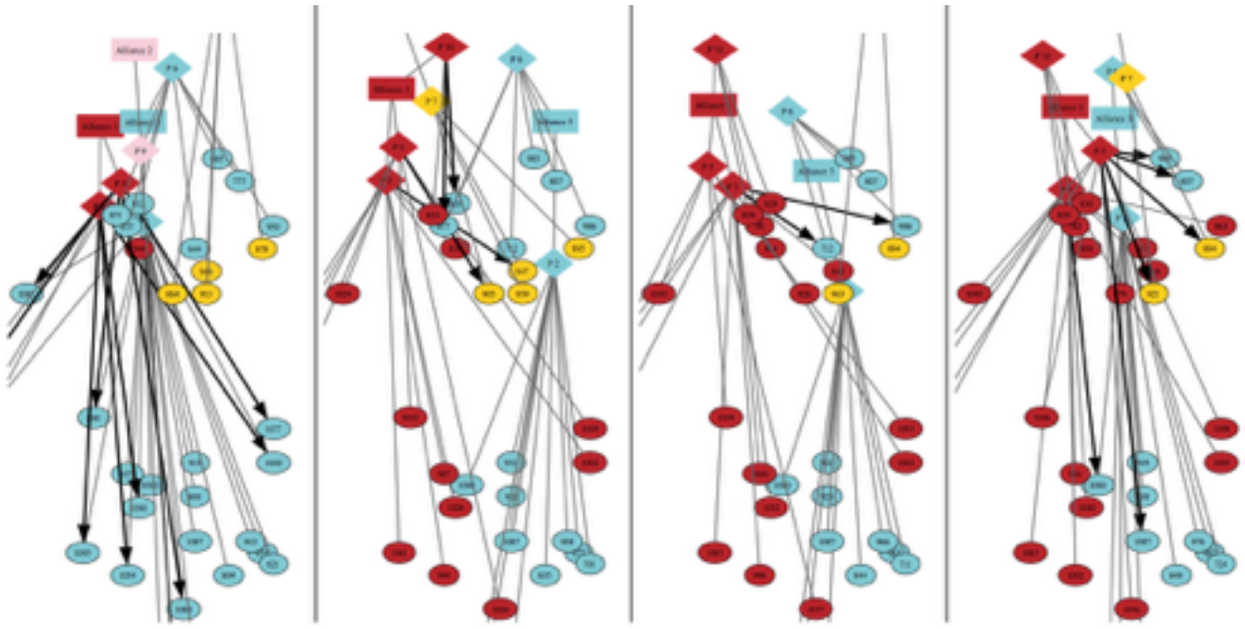
how does the world change over time?

```
0.4::conquest(Attacker,C); 0.6::nil :-
    city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

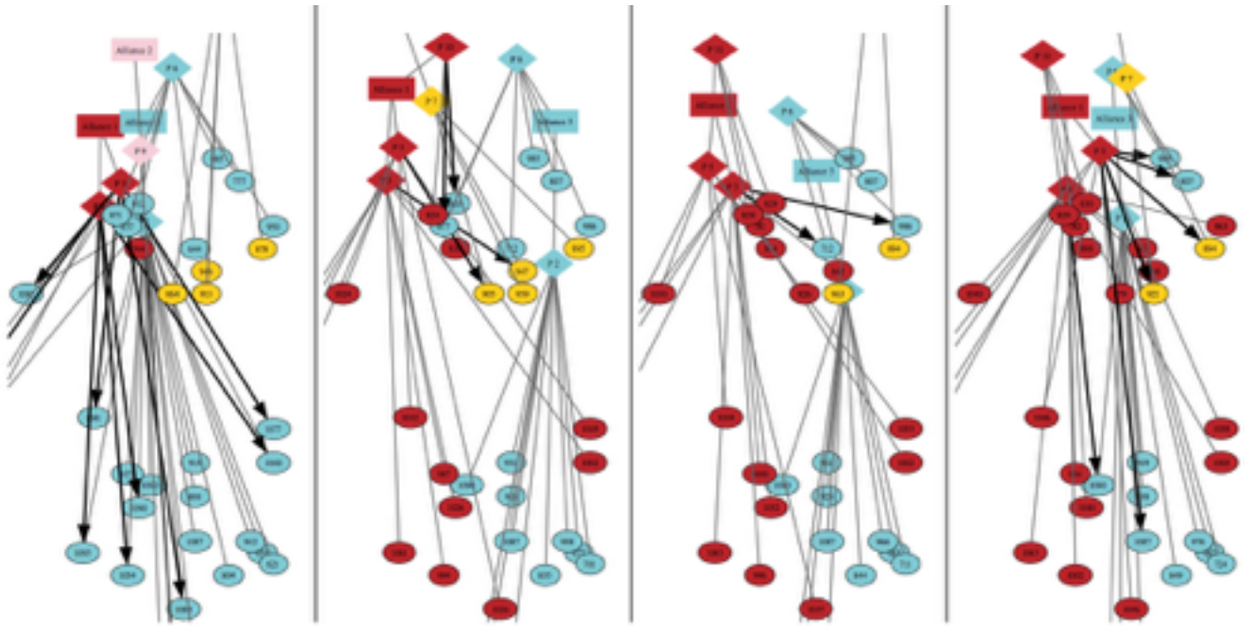
one of the **effects** holds at time T+1

```
0.4::conquest(Attacker,C); 0.6::nil :-
        city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

one of the **effects** holds at time T+1

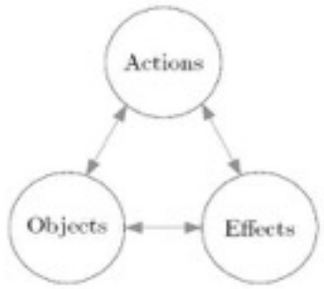
```
0.4::conquest(Attacker,C); 0.6::nil :-
                    city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Learning relational affordances

Learn probabilistic model



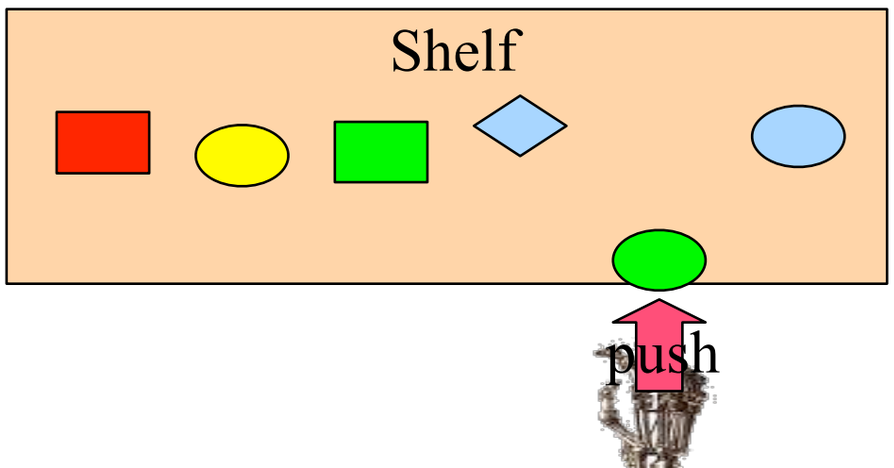
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

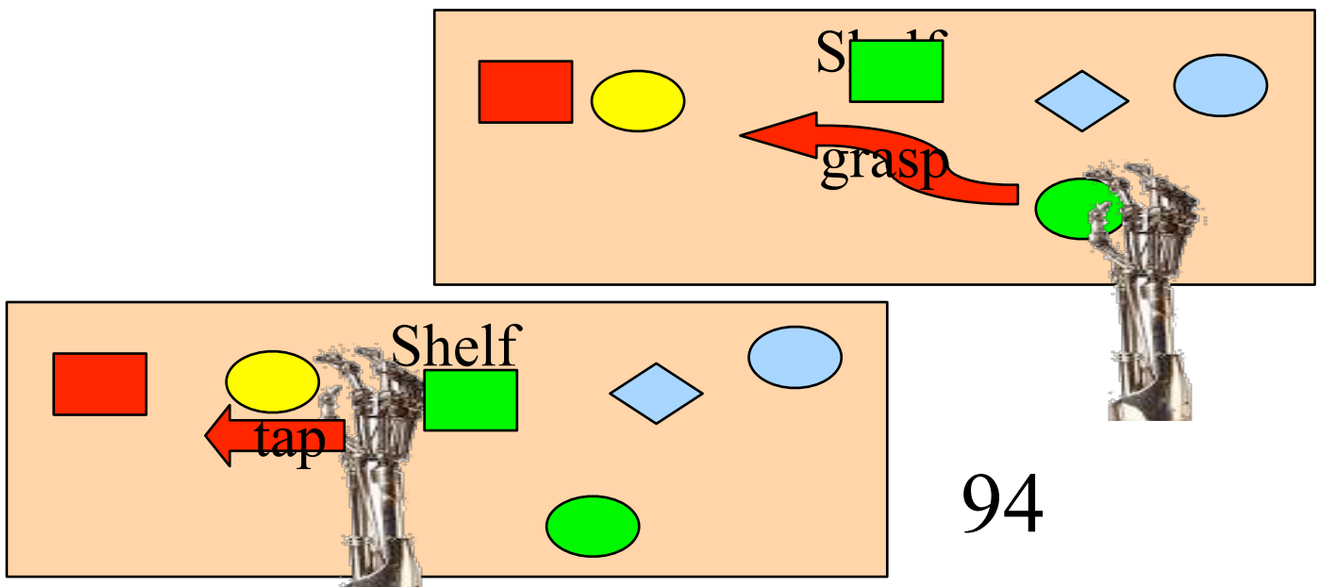
Learning relational affordances between two objects (learnt by experience)

From two object interactions

Generalize to N

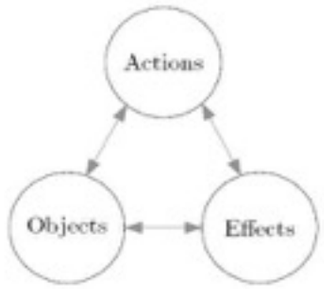
Moldovan et al. ICRA 12, 13, 14





Learning relational affordances

Learn probabilistic model



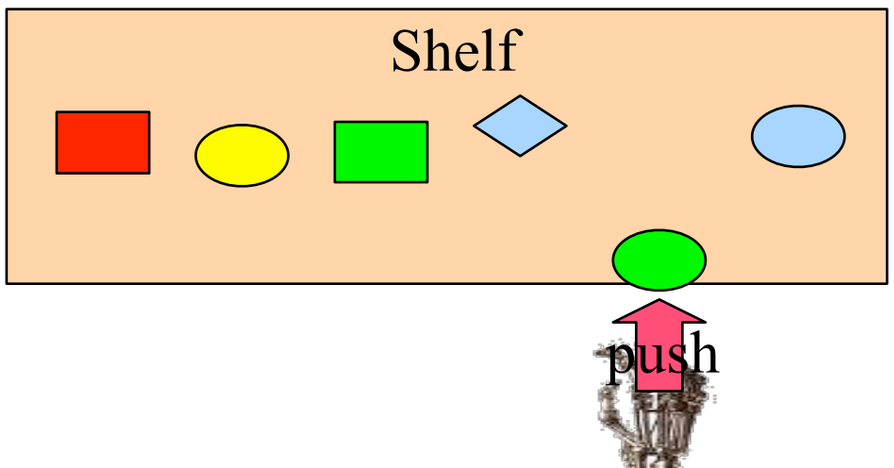
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

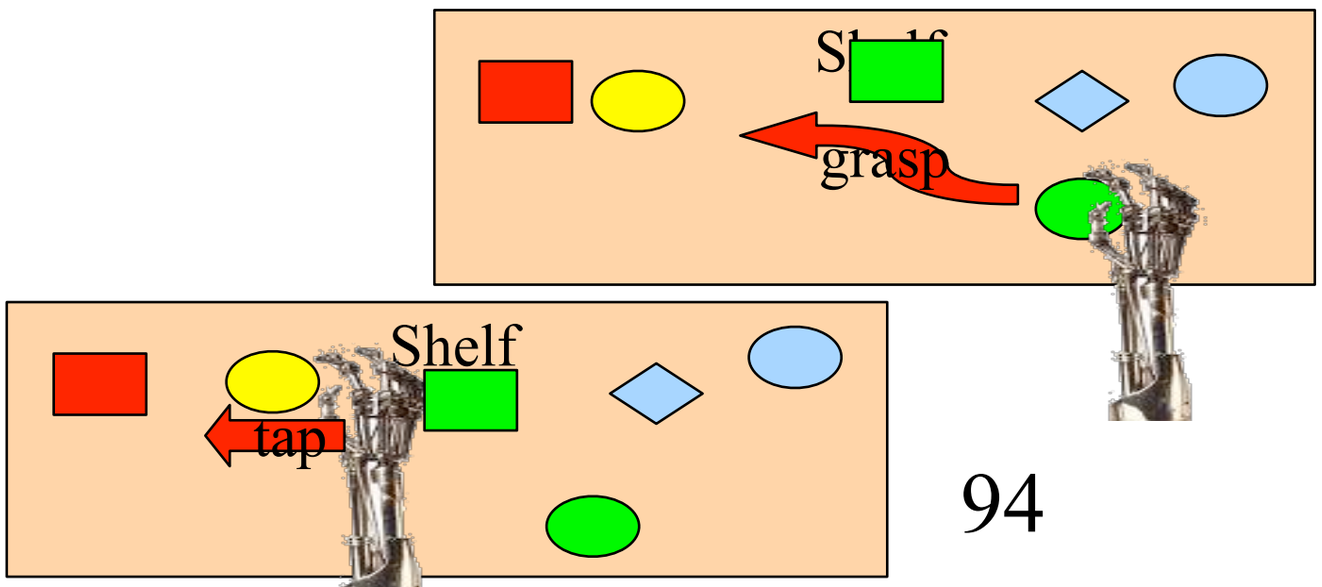
Learning relational affordances between two objects (learnt by experience)

From two object interactions

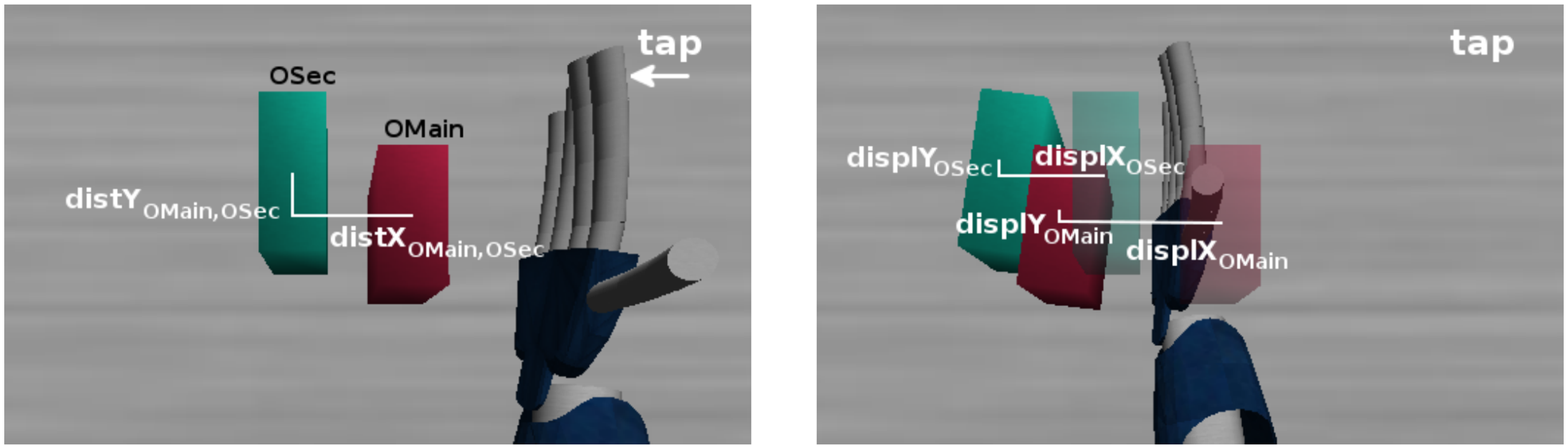
Generalize to N

Moldovan et al. ICRA 12, 13, 14





What is an affordance ?



Clip 8: Relational O before (l), and E after the action execution (r).

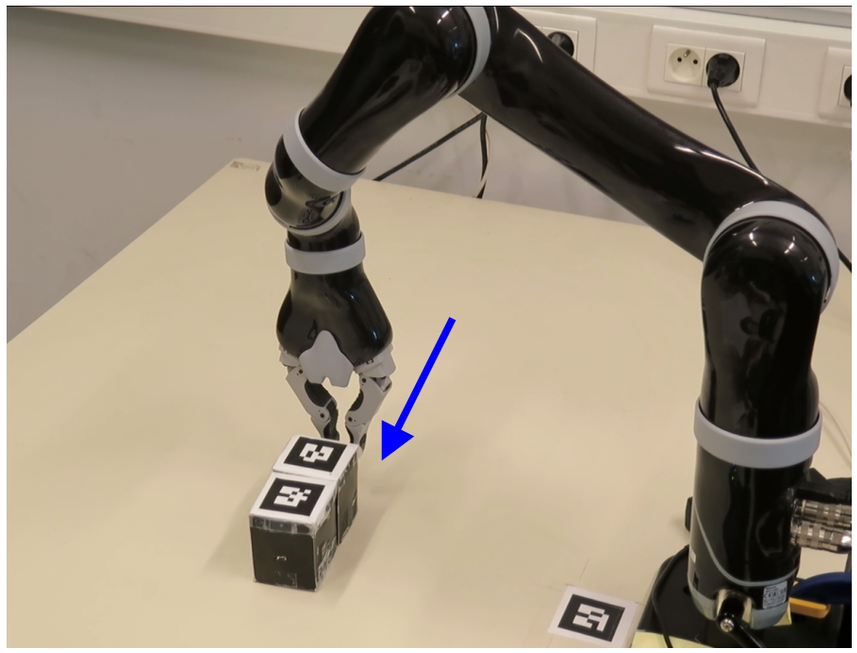
Table 1: Example collected O, A, E data for action in Figure 8

Object Properties	**Action**	**Effects**
$shape_{O_{Main}} : sprism$ $shape_{O_{Sec}} : sprism$ $distX_{O_{Main},O_{Sec}} : 6.94cm$ $distY_{O_{Main},O_{Sec}} : 1.90cm$	$tap(10)$	$displX_{O_{Main}} : 10.33cm$ $displY_{O_{Main}} : -0.68cm$ $displX_{O_{Sec}} : 7.43cm$ $displY_{O_{Sec}} : -1.31cm$

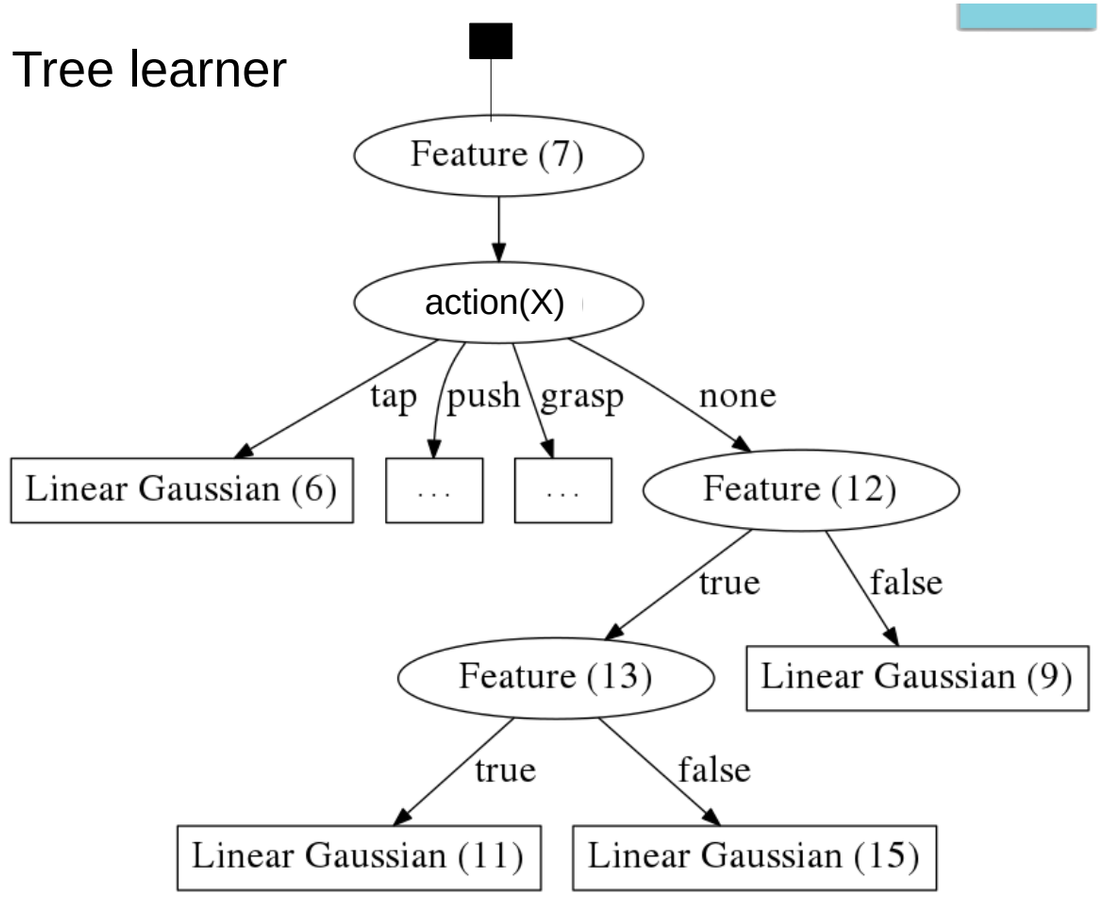
- Formalism — related to STRIPS but models delta
- but also joint probability model over A, E, O

Relational Affordance Learning

- **Learning the Structure of Dynamic Hybrid Relational Models**
  Nitti, Ravkic, et al. ECAI 2016
  - Captures relations/affordances
  - Suited to learn affordances in
    robotics set-up, continuous and discrete variables
  - Planning in hybrid robotics domain



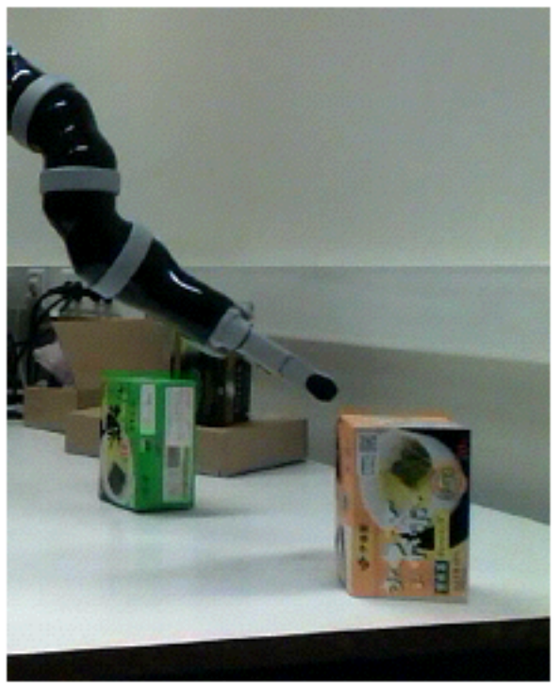
DDC Tree learner

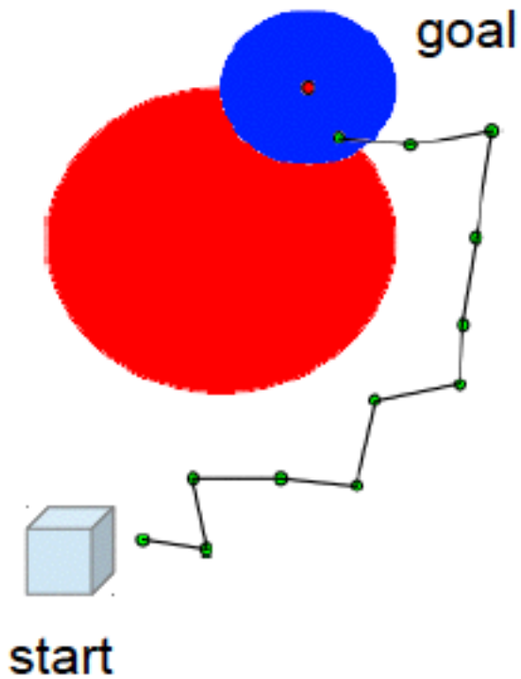


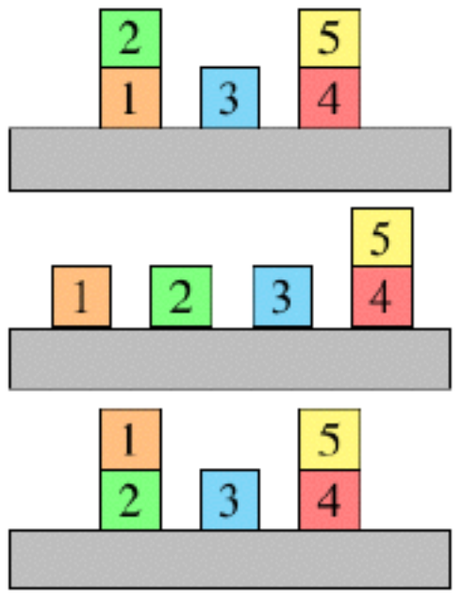
Planning

- Main task: probabilistic planning

  Find the best action to achieve the goal
- Discrete + continuous + relational representation







[Nitti et al ECML 15; MLJ 17]

ProbLog for activity recognition from video

CAVIAR-INRIA human activity dataset

28 videos
≈ 26.500 frames

- Separation between low-level events (LLE) and high-level events (HLE)
  - LLE: *walking, running, active, inactive, abrupt*
  - HLE: *meeting, moving, fighting, leaving_object*
- Probabilistic Logic approach: *Event Calculus in ProbLog* (Prob-EC) to infer the high-level events from an **algebra** of low-level events.
- Example:

$$initiatedAt(fighting(P_1, P_2) = true, T) \leftarrow$$
$$happensAt(abrupt(P_1), T),$$
$$holdsAt(close(P_1, P_2, 44) = true, T),$$
$$not\ happensAt(inactive(P_2), T).$$

[Skarlatidis et al, TPLP 13]

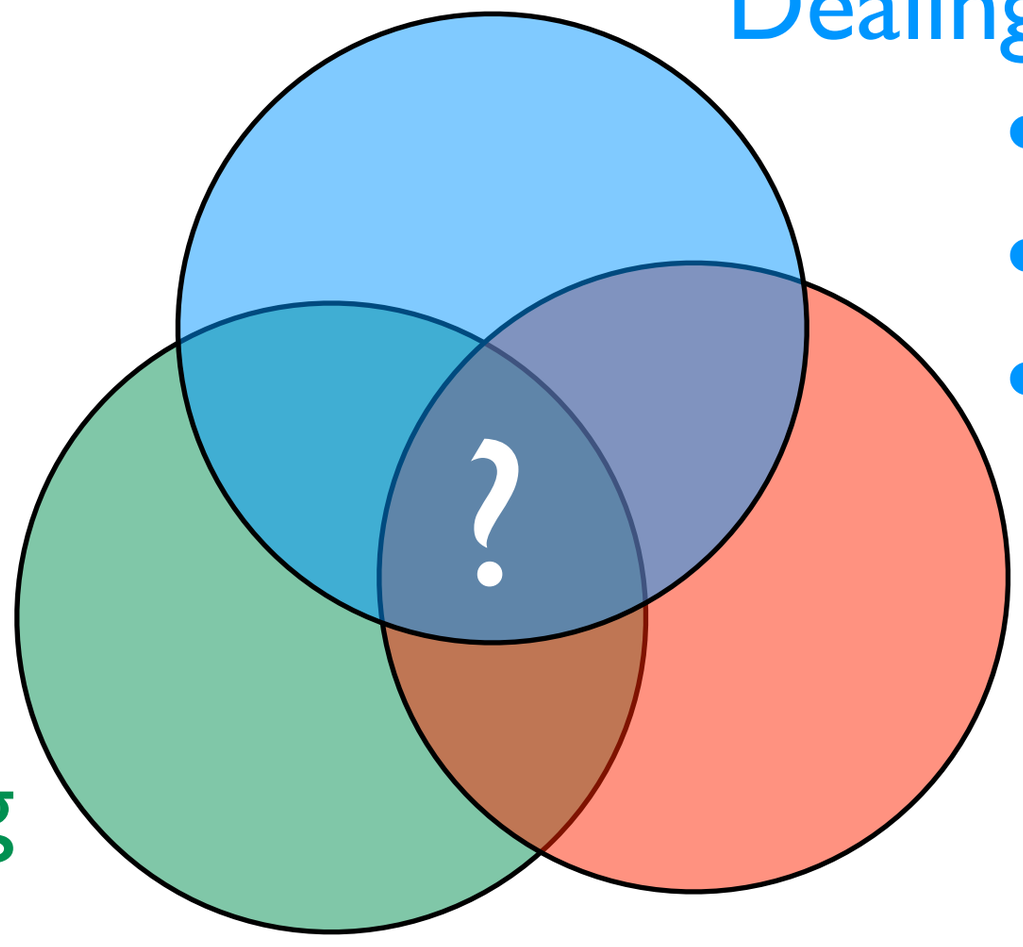
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...

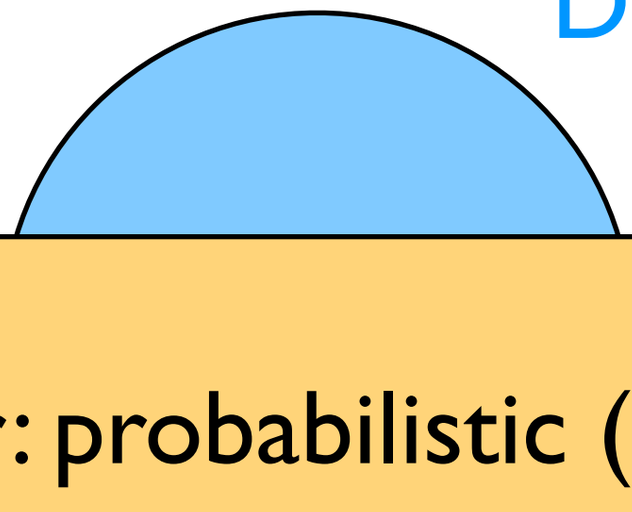


Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

A key question in AI:



Dealing with uncertainty
- probability theory

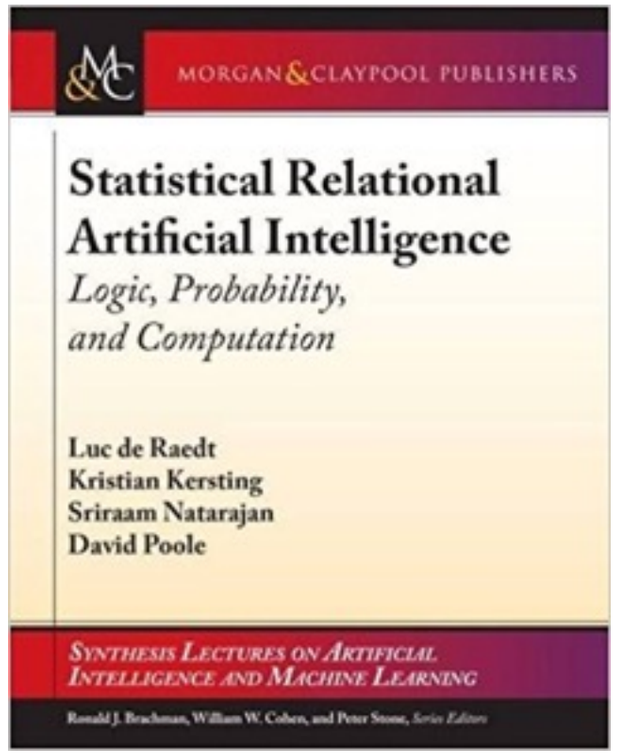
Reas
relat
- log
- da
- pr
- ...

- Our answer: probabilistic (logic) programming
  = probabilistic choices + (logic) program
- Many languages, systems, applications, ...
- ... and much more to do!

structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Further Reading

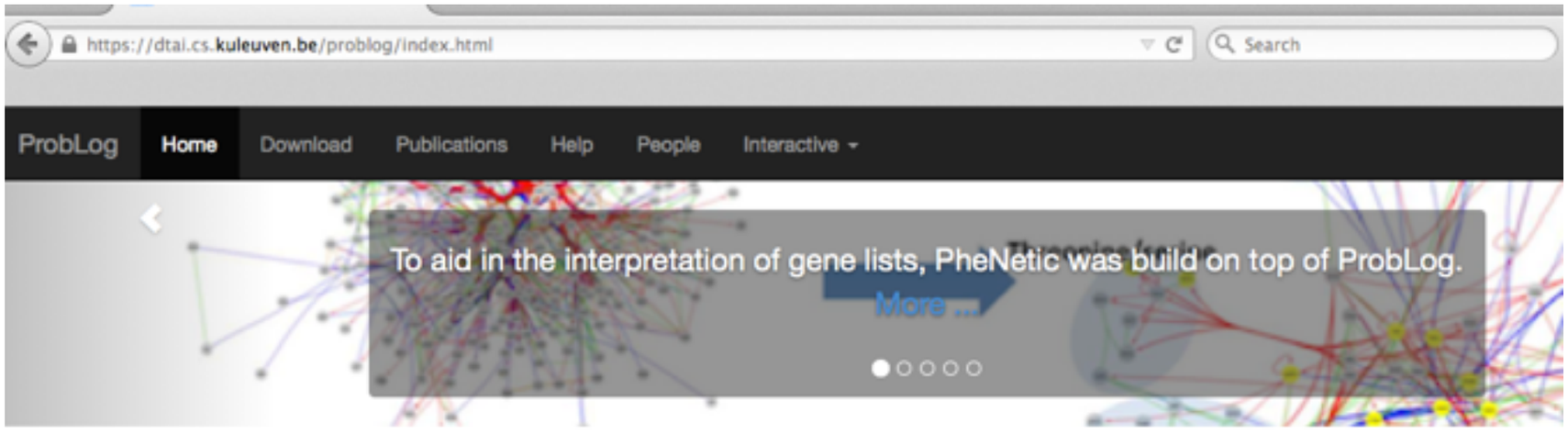


- One book
- Three websites to start
  - http://probmods.org/ Probabilistic Models of Cognition — Church
  - http://dtai.cs.kuleuven.be/problog/ — check also [DR & Kimmig, MLJ 15]
  - http://alchemy.cs.washington.edu/ —Markov Logic, check also [Domingos & Lowd] Markov Logic, Morgan Claypool.

Maurice Bruynooghe
Bart Demoen
Anton Dries
Daan Fierens
Jason Filippou
Bernd Gutmann
Manfred Jaeger
Gerda Janssens
Kristian Kersting
Angelika Kimmig
Theofrastos Mantadelis
Wannes Meert
Bogdan Moldovan
Siegfried Nijssen
Davide Nitti
Joris Renkens
Kate Revoredo
Ricardo Rocha
Vitor Santos Costa
Dimitar Shterionov
Ingo Thon
Hannu Toivonen
Guy Van den Broeck
Mathias Verbeke
Jonas Vlasselaer

Thanks !

http://dtai.cs.kuleuven.be/problog



PLP Systems

- **PRISM** http://sato-www.cs.titech.ac.jp/prism/
- **ProbLog2** http://dtai.cs.kuleuven.be/problog/
- Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  - **ProbLog1**
  - **cplint** https://sites.google.com/a/unife.it/ml/cplint
  - **CLP(BN)**
  - **LP2**
- **PITA** in XSB Prolog http://xsb.sourceforge.net/
- **AILog2** http://artint.info/code/ailog/ailog2.html
- **SLPs** http://stoics.org.uk/~nicos/sware/pepl
- **contdist** http://www.cs.sunysb.edu/~cram/contdist/
- **DC** https://code.google.com/p/distributional-clauses
- **WFOMC** http://dtai.cs.kuleuven.be/ml/systems/wfomc